Escola Politècnica Superior
d'Enginyeria de Vilanova i la Geltrú

UNIVERSITAT POLITÈCNICA DE CATALUNYA

PROJECTE FI DE CARRERA

TÍTOL: Simulación y control de procesos automatizados con PLCs OMRON y LabVIEW.

AUTOR: Daniel Owono Marti

TITULACIÓ: Enginyeria Tècnica Industrial Esp. Electrònica Industrial

DIRECTOR: Cristóbal Raya Gine

DEPARTAMENT: ENGINYERIA ELECTRÒNICA

DATA: 27 de enero de 2011

TÍTOL: Simulación y control de procesos automatizados con PLCs OMRON y LabVIEW

COGNOMS: Owono Marti **NOM: Daniel**

TITULACIÓ: Enginyeria Tècnica Industrial

ESPECIALITAT: Electrònica Industrial **PLA: 95**

DIRECTOR: Cristóbal Raya Gine
DEPARTAMENT: ENGINYERIA ELECTRÓNICA

QUALIFICACIÓ DEL PFC

TRIBUNAL

PRESIDENT	SECRETARI	VOCAL
Pedro Ponsa Asensio	**Mariano Lopez Garcia**	**Pau Marti Colom**

DATA DE LECTURA:

08/02/2011

Aquest Projecte té en compte aspectes mediambientals: ***Sí*** *No*

PROJECTE FI DE CARRERA

RESUM (màxim 50 línies)

Con los nuevos grados instaurados por el nuevo plan universitario, hemos creado unas prácticas de automatización industrial gestionadas con los autómatas de OMRON que la facultad acaba de adquirir. Estos PLCs permiten trabajar con nuevas técnicas de programación y, además tienen incorporado más módulos que permite un aprendizaje más amplio sobre la materia.

Para ello, he realizado unas prácticas diseñadas bajo el programa de LabVIEW, con el propósito de simular unas plantas reales automatizadas. Con lo cual, el alumno puede adquirir mejores conocimientos al interactuar con mayor realismo sobre un proceso automatizado.

La comunicación entre el autómata y el programa LabVIEW se gestiona con el motor de comunicación CX-Server Lite de OMRON.

LabView	PLC	OMRON	CX-Server Lite
CX-Programmer	Cx-Simulator	Practicas	

Paraules clau (màxim 10)

SUMARIO

CAPÍTULO 1

En este primer capítulo, se ha expuesto una breve introducción, explicando las necesidades y motivaciones del proyecto de este proyecto así como los objetivos propuestos que queremos alcanzar. También se dedican unas líneas detallando el hardware y el software utilizados.

CAPÍTULO 2

En el segundo capítulo, se encuentra una pequeña introducción sobre los autómatas programables. Por consiguiente, exponemos las características del PLC que empleamos. También se describe los lenguajes de programación que se usan en las prácticas.

CAPÍTULO 3

En el tercer capítulo, se explica genéricamente el programa que hemos utilizado para diseñar las plantas, LabVIEW. Consecutivamente hemos hecho una breve exposición sobre el protocolo de comunicación usado, CX-Server pero en nuestro caso bajo la tecnología ActiveX que es el CX-Server Lite. De igual forma, comentamos la evolución de las librerías de comunicación que nos brinda El ActiveX de OMRON.

CAPÍTULO 4

En el cuarto capítulo, se han detallado todas las prácticas que hemos diseñado para los ejercicios de automatización. Asimismo hemos especificado la forma en que hemos programado las prácticas en LabVIEW, y también puntualizando las partes de programación mas complejas de cada ejercicio.

CAPÍTULO 5

En el capítulo cinco, nos centramos en como se configura el programador de OMRON para programar el PLC. Asimismo, también explicamos como se configura el CX-Simulator. Por ultimo planteo los objetivos de programación de las prácticas de sistemas de automatización a realizar.

CAPÍTULO 7

En el capítulo séptimo, plasmo unas reflexiones sobre el trabajo realizado y las conclusiones a las que se han llegado.

BIBLIOGRAFIA

En este apartado, hemos confeccionado una lista de los manuales y paginas web que hemos utilizado para diseñar las prácticas.

ANEXO

En esta sección, se explica la organización de los documentos adjuntos del proyecto y, especificaciones sobre los métodos de LabVIEW utilizados para diseñar la comunicación.

CAPITULO 1. INTRODUCCIÓN

En este capitulo exponemos las necesidades y motivaciones de este proyecto así, como los objetivos propuestos que se quieren alcanzar. También se dedican unas líneas a detallar del hardware y el software utilizados.

1.1. INTRODUCCION

Este proyecto está orientado para realizar prácticas de automatización industrial con PLCs de OMRON de la serie CJ1M.

1.2. ORIGEN DEL PROYECTO

En la creación de nuevos grados en la facultad la escuela EPSEVG ha adquirido nuevos PLCs de OMRON CJ1M, los cuales permite trabajar con nuevas técnicas de programación “grafcet” sfc, bloques de funciones, texto estructurado y además estos nuevos PLCs de OMRON tienen módulos de comunicación ETHERNET cosa que los anteriores PLC de SIEMENS de la serie 200 que estaban en el laboratorio no tenían.
Hasta ahora las prácticas del laboratorio simulaban las entradas con interruptores y las salidas con leds. Entonces era difícil entender exactamente el funcionamiento de una planta automatizada, y también costaba depurar los programas.

1.3. OBJETIVOS DEL PROYECTO

El proyecto esta fundamentado en el desarrollo de unas prácticas basadas en la automatización industrial. Dichas prácticas serán estructuradas para que el estudiante pueda interactuar con la planta como si fuera una planta real, así no tendrá que imaginar su funcionamiento, por lo tanto el usuario puede programar el controlador para que pueda gestionar la planta implementada en la práctica.
Las practicas se diseñaran bajo el programa LabVIEW, es decir el programa LabVIEW es quien hace de planta de procesos y juntamente dicha planta es controlada por un PLC de OMRON.
La comunicación entre el PLC y el programa de LabVIEW se gestiona con el motor de comunicación CX-Sever Lite. Dicho motor son librerías que nos da la marca Omron para la comunicación entre el PLC y el PC.

1.4. SOFTWARE Y HARDWARE UTILIZADOS

El programa principal utilizado es el LABVIEW, versión 2009, dicho programa es el encargado de simular una planta real en cada práctica. También se han empleado programas de diseño gráfico para la creación de imágenes que simulan una planta real como el PHOTOSHOP, FREEHAND y, COREL SHOP PRO. Y por ultimo para diseñar la comunicación se ha usado en paquete CX-Server lite de OMRON.

El desarrollo y el ámbito de ejecución son mediante Windows XP. El ordenador consta de un AMD sempron (tm) Processor 3100 con una potencia de 1,8 GHz y una capacidad de tarjetas RAM de 940 MB.
El programa para implementar el Ladder y SFC es el CX-PROGRAMMER de OMRON, versión 7.2 el cual está asociado al paquete CX-ONE junto con otras aplicaciones de interés donde también esta las librerías de gestión y control de datos CX-Server Lite.

CAPITULO 2. CONTROLADOR LÓGICO PROGRAMABLE (PLC)

En el presente capitulo hago una descripción genérica de los autómatas programables como los tipos de lenguajes que se usan a la hora de programarlos. Asimismo exponemos las características del PLC utilizado en el proyecto.

2.1. INTRODUCCION

Un autómata programable industrial (API) o Programable logic controller (PLC), es un equipo electrónico, programable en lenguaje no informático, diseñado para controlar en tiempo real y en ambiente del tipo industrial, procesos secuenciales.
Un PLC trabaja en base a la información recibida por los sensores y el programa lógico interno, actuando sobre los actuadores de la instalación.
Uno de sus propósitos fue sustituir a los circuitos eléctricos de lógica cableada, la cual es una forma de realizar controles, en la que el tratamiento de datos (botonería, fines de carrera, sensores, presostatos, etc.), se efectúa en conjunto con contactares o relés auxiliares, frecuentemente asociados a temporizadores y contadores.

2.1.1. Ventajas del PLC respecto de la lógica cableada

- El mecanismo es de carácter estándar por que la variedad de los componentes que componen es mínima, posibilidad de ampliación y/o modificación del sistema mediantes la sustitución o agregando módulos.
- En el caso de eliminación de una maquina, el sistema de control se reutiliza en tras aplicaciones.
- Pueden incorporarse a la maquina el cual esta funcionando.
- Se puede modificar el programa con el sistema funcionando, por lo tanto, permite una óptima adaptación al proceso.
- Tiene interfaz de comunicación, impresora y otros periféricos.
- No es necesario dibujar el esquema de contactos.
- No es necesario simplificar las ecuaciones lógicas, ya que, generalmente, la capacidad de almacenamiento del modulo es suficientemente grande.
- Existen módulos de comunicación para el PLC.
- Robusto y esta diseñado para trabajar en condiciones ambientales complejas.

2.2. ESTRUCTURA DE UN PLC

El PLC es un tipo de ordenador especialmente diseñado para un entorno industrial, para ocupar el lugar de la unidad que gobierna el proceso productivo. La estructura interna del PLC se basa en módulos que están conectados mediante un bus interno. Principalmente consta de dos partes fundamentales: el hardware y el software.

2.2.1. Características del hardware

El PLC se compone esencialmente de algunas partes comunes a todos los modelos, y otras que dependen de la envergadura del mismo y la aplicación donde se usará.

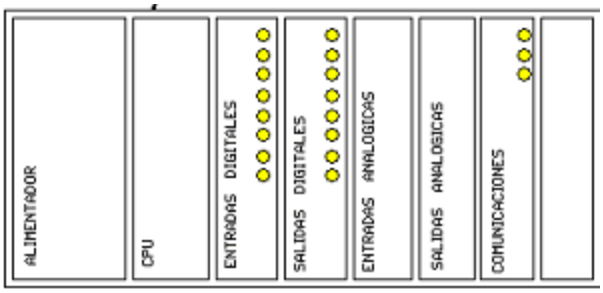


Fig. 1. Croquis de un PLC.

- **Fuente de alimentación**: Es la que da energía a todo el conjunto de módulos y también puede alimentar los dispositivos de entrada. La tensión de alimentación puede varias de 5- 24 V.

- **CPU (Unidad central)**: Es la que procesa toda la información recibida del exterior y activa la señal de las salidas, una vez procesada la programación.

- **Memoria**: Es donde se almacena los códigos encargados para ejecutar el programa.

- **Módulo de entrada**: Es donde están conectados los sensores y detectores que controlan el proceso.

- **Módulo de salida**: Es donde están conectados los actuadores del sistema y recibe la información de la CPU mediante bus interno.

- **Módulos especiales**: Estos se pueden acoplar al bus. (módulo de E/S analógica).

Los módulos descritos son instalados en una rack que disponen tantos slots como módulos a montar. Los racks se comunican con la CPU mediante buses y pueden configurarse en local o remotamente. Los locales se comunican con la CPU en paralelo o en serie RS-232, cubriendo distancias de algunos metros. Los remotos en cambio se comunican en modo ETHERNET o serie RS-485, pueden cubrir varios kilómetros.

2.2.2. Tipos de autómatas

Podemos identificar dos tipos de autómatas de acuerdo con su estructura pueden ser compactos o modulares:

- En el primer caso las interfaces de E/S son limitadas y el autómata no permite expansiones; generalmente son dispositivos de bajo coste.

- Para el segundo caso, el autómata admite la configuración del hardware que este disponible para su gama de productos correspondientes, y puede ser reconfigurado mediante la incorporación o eliminación de módulos.

2.2.3. Memoria interna

En un autómata programable, la memoria interna es una RAM donde almacena el estado de las variables que maneja el autómata: entradas, salidas, contadores, relés internos, señales de estado, etc. Esta memoria interna se encuentra dividida en varias áreas, cada una de ellas con un cometido y características distintas.

Existen diferentes tipos de tecnologías aplicadas a los autómatas, las mas utilizadas son:

- **ROM (Read Only Memory)** o memorias solo de lectura. La escritura de la información se lleva a termino durante la construcción por la cual el contenido no se puede modificar ni borrar.

- **PROM (Programmable Read Only Memory),** son solo de lectura pero programable por el usuario antes de ser utilizados. Una vez programados son
- inalterables.

- **EPROM (Erase Programmable Read Only Memory)** son solo de lectura pero reprogramable por el usuario el cual previamente ha cancelado la información anterior por medio de radiación ultraviolada, que incide al chip por una ventana.

- **EEPROM (Electrically Erase PROM),** ROM programable y se puede borrar eléctricamente.

- Es un tipo de memoria ROM que puede ser programado, borrado y reprogramado eléctricamente, a diferencia de la EPROM. Son memorias no volátiles que poden ser **l**eídas un numero ilimitado de veces, pero solo puede ser borrado y reprogramado entre 100.000 i un millón de veces.

2.3. LENGUAJES DE PROGRAMACION

Los llenguatges de programación son necesarios para la comunicación entre el usuario y la máquina o proceso donde se encuentre el PLC. La interacción que tiene

el usuario con el PLC las pueden hacer mediante la utilización de un cargador de programa también reconocida como una consola de programación o a través de un PC.
En procesos grandes o en ambientes industriales el PLC recibe el nombre de API (Autómata Programable Industrial) y utiliza como interface para al usuario pantallas de plasma, pantallas de contacte (touch screen) o sistemas SCADA (sistemas para la adquisición de datos, supervisión, monitorización y control de los procesos.

2.3.1. Tipos de lenguaje de programación

Los lenguajes de programación para un PLC son de dos tipos: visuales y escritos

Visuales

Se basan en estructurar el programa por medio de símbolos gráficos, similares a los se usan para describir los sistemas de automatización, planos esquemáticos y diagramas de bloques. Es un lenguaje de alto nivel y se pueden distinguir diversos:

- Diagrama de contactos: viene ser funcionalmente el proceso. Se representa como un circuito de contactos y relés, fácil de entender y de utilizar para a usuarios con experiencia en lógica cableada.

  Este lenguaje se define como LADDER (Escalera) por que la forma de construcción del esquema se parece a una escalera.

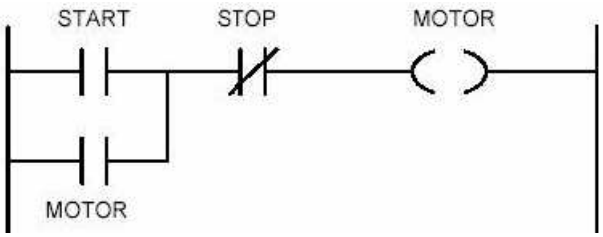


Fig. 2. Programación Ladder.

- Diagrama de bloques funcionales: Utiliza los diagramas lógicos de la electrónica digital. Los bloques funcionales son los equivalentes de los circuitos integrados que representan funciones de control especializados. Tienen una interface de entradas y salidas claramente definidas y un código interno oculto, como un circuito integrado.

  De esta manera, se establece una clara separación entre los diferentes niveles de programadores, o el personal de mantenimiento. Una vez definido, puede ser usado una y otra vez, cosa que lo hace reutilizable.

Fig. 3. Programación con bloques funcionales.

- Organigrama de bloques secuenciales: Representa la versión que todo proceso cumple una secuencia. Este lenguaje son los mas utilizados por los programadores de PLC con mas frecuencia.

  Un ejemplo es el GRAFCET, el cual es un diagrama funcional normalizado, que permite hacer un modelo de proceso a automatizar, contemplan entradas, acciones a realizar, y los procesos intermedios que provocan estas acciones. Es un potente lenguaje grafico de programación para PLCs, adaptado a la resolución de sistemas secuenciales. Sirve para elaborar el modelo pensando en la ejecución directa del automatismo o programa de autómata.

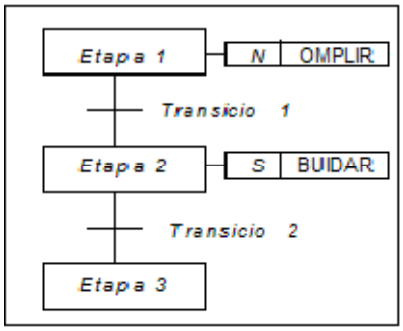


Fig. 4. grafcet.

Escritos

Son listados de sentencias que describen las funciones a ejecutar instrucción a instrucción. Normalmente se caracteriza por ser un lenguaje de bajo nivel, similar al lenguaje ensamblador, con una sintaxis y vocabulario de acuerdo con la terminología utilizada en PLC, aunque hay que son de alto nivel. Tenemos dos tipos claramente definidos:

- **Lista de instrucciones**: Son instrucciones del tipo booleano, utiliza representaciones de letras y números. Se usan abreviaturas mnemotécnicas y no requiere gran memoria para los trabajos de automatización. Una desventaja es la cantidad de trabajo necesario para la programación de un proceso que necesita un programa con muchas instrucciones.

Siemens	Telemecanique	General Electric
U E0.1	L I0.01	LD %I0001
U E0.2	A I0.02	AND %I0002
O E0.3	O I0.03	OR %I0003
= A3.1	= O3.01	OUT %Q0031

- **Texto estructurado**: Es un lenguaje del tipo booleano de alto nivel y estructurado que incluye las sentencias de selección (IF-THEN-ELSE) y de iteración (FOR, WHILE Y REPEAT), a de mas otras funciones especificas para aplicaciones de control.

  Su uso óptimo es para aplicaciones en las que es necesario realizar cálculos matemáticos, comparativos, etc.

2.4. AUTOMATA CJ1M-22 OMRON

Este modelo de autómata es el que se utilizara en las prácticas de automatización. A continuación vamos a explicar sus aspectos mas importantes:

2.4.1. Características

Los PLCs CJ1M-22 es un autómata del fabricante de OMRON que dispone de mucha versatilidad. A demás de tener una medida reducida en comparación con sus predecesores, también cuenta con una capacidad de proceso realmente rápida donde las instrucciones básicas son ejecutadas a 0.02 us y las especiales a 0.06 us. Es un dispositivo modular y se coloca sobre raíles DIN. La capacidad de programación llega hasta 250 pasos. Respecto a las comunicaciones dispone de la posibilidad de manejar DeviceNet, Controller Link y Ethernet.

Para la parte de programación, y mediante el software CX-Programmer, el programa completo se puede dividir en tareas que controlan procesos, sistemas de control y funciones independientes, por lo tanto, se puede desarrollar simultáneamente por diversos programadores.

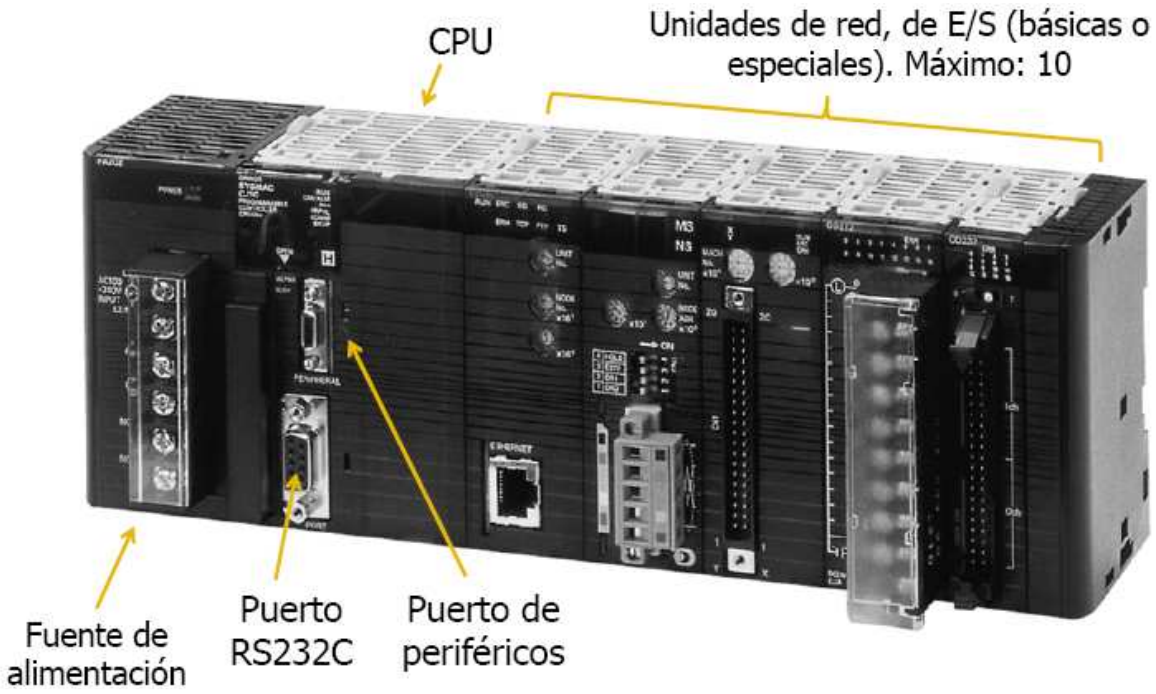


Fig. 5. PLC de OMRON CJ1M CPU-22.

El CJ1M-22 dispone de funciones de comunicación con ordenadores personales, y otros autómatas de OMRON y terminales programables OMRON.

Entre las funciones que dispone tenemos:

- Transferencia de datos o de tarjetas de memoria
- Transferencia automática de archivos cuando arranca
- Sustitución del programa durante la operación
- Instrucciones de bucle
- Instrucciones de pila
- Instrucciones de tabla de registros
- Entradas de interrupción de alta velocidad
- Contadores de alta velocidad

Elemento		Especificación					
		CPUs con E/S incorporadas			CPUs sin E/S incorporadas		
Modelo		CJ1M-CPU23	CJ1M-CPU22	CJ1M-CPU21	CJ1M-CPU13	CJ1M-CPU12	CJ1M-CPU11
Puntos de E/S		640	320	160	640	320	160
Memoria de programa de usuario		20 Kpasos	10 Kpasos	5 Kpasos	20 Kpasos	10 Kpasos	5 Kpasos
Número máximo de bastidores expansores		1 máx.	No compatible.		1 máx.	No compatible.	
Memoria de datos		32 Kcanales					
Memoria de datos extendida		No compatible.					
Tiempo de inicio de salida de impulsos		• 46 µs (sin aceleración/deceleración) • 70 µs (con aceleración/deceleración)		• 63 µs (sin aceleración/deceleración) • 100 µs (con aceleración/deceleración)	—		
Entradas de interrupción		2		1	2		1
Puntos de salida PWM		2		1	Ninguno		
Número máximo de subrutinas		1.024		256	1.024		256
Número máximo de saltos para la instrucción JMP		1.024		256	1.024		256
Entradas incorporadas		10 • Entradas de interrupción (respuesta rápida): 4 entradas • Contador de alta velocidad: 2 entradas (fase diferencial a 50 kHz o monofásica a 100 kHz)			—		
Salidas incorporadas		6 • Salidas de impulsos: 2 a 100 kHz • Salidas PWM: 2		6 • Salidas de impulsos: 2 a 100 kHz • Salidas PWM: 1	—		
Bloques de funciones	Nº máximo de definiciones	128					
	Nº máximo de instancias	256					
Memoria flash	Memoria de programas FB (Kbytes)	256					
	Archivos de comentarios (Kbytes)	64					
	Archivos de índices de programas (Kbytes)	64					
	Tablas de símbolos (Kbytes)	64					
Consumo eléctrico (suministrado por unidades de fuente de alimentación)		0,64 A a 5 Vc.c.			0,58 A a 5 Vc.c.		
Conector (incluido)		La CPU incluye de serie un conector RS-232C (enchufe: XM2A-0901; carcasa XM2S-0911-E)					

Tabla 1. Modelos de OMRON CJ1M.

2.4.2. Dispositivos básicos

A continuación expondremos los diferentes dispositivos del autómata de OMRON.

CPU

Las CPUs CJ1M son PLC avanzados de alta velocidad y tamaño micro equipados con E/S incorporada. Las E/S incorporadas tienen las siguientes características.
Las entradas y salidas incorporadas de la CPU se pueden utilizar como entradas y salidas de empleo general. En especial, se puede realizar el refresco inmediato de la E/S en mitad de un ciclo del PLC al ejecutar una instrucción para ello.

Modelo	Numero de I/O bits	Numero máximo de racks de expansión	Máximo numero de unidades conectables	capacidad de programa	Capacidad memoria de datos	Tiempos de proceso de instrucciones	Puertos de entrada	Opciones montables	I/O
CJ1M-CPU12	320 puntos	ninguno	10 unidades	10 ksteps	32 kwords(DM only. No EM)	100 ns	Puerto Periferico y puerto rs 232	100 ns memoria flash	Ninguno
CJ1M-CPU13	640 puntos	1 unidad	CPU rack: 10 unidades Expansión rack:10 unidades	10 ksteps					10 entradas y 6 salidas Entradas: 4 entradas de interrupción; 2 entradas de contador de alta velocidad (Fase diferencial: 50 kHz; Monofásica: 100 kHz) Salidas: 2 salidas de pulsos (2 puntos para posicionado, control de velocidad de 100-kHz, y salidas PWM):
CJ1M-CPU22	320 puntos	ninguno	10 unidades	10 ksteps					
CJ1M-CPU23	640 puntos	1 unidad	CPU rack: 10 unidades Expansión rack:10 unidades	10 ksteps					

Tabla 2. Modelos del CJ1M.

Unidades de expansión

Las unidades de expansión son típicas de los PLCs modulares los cuales puedes montar según las características del sistema que quieras controlar. Las expansiones se colocan después de la CPU del PLC a controlar mediante unas grapas se cogen unas a otras como un tren.

Hay diversas expansiones según el proceso a tratar, los cuales son:

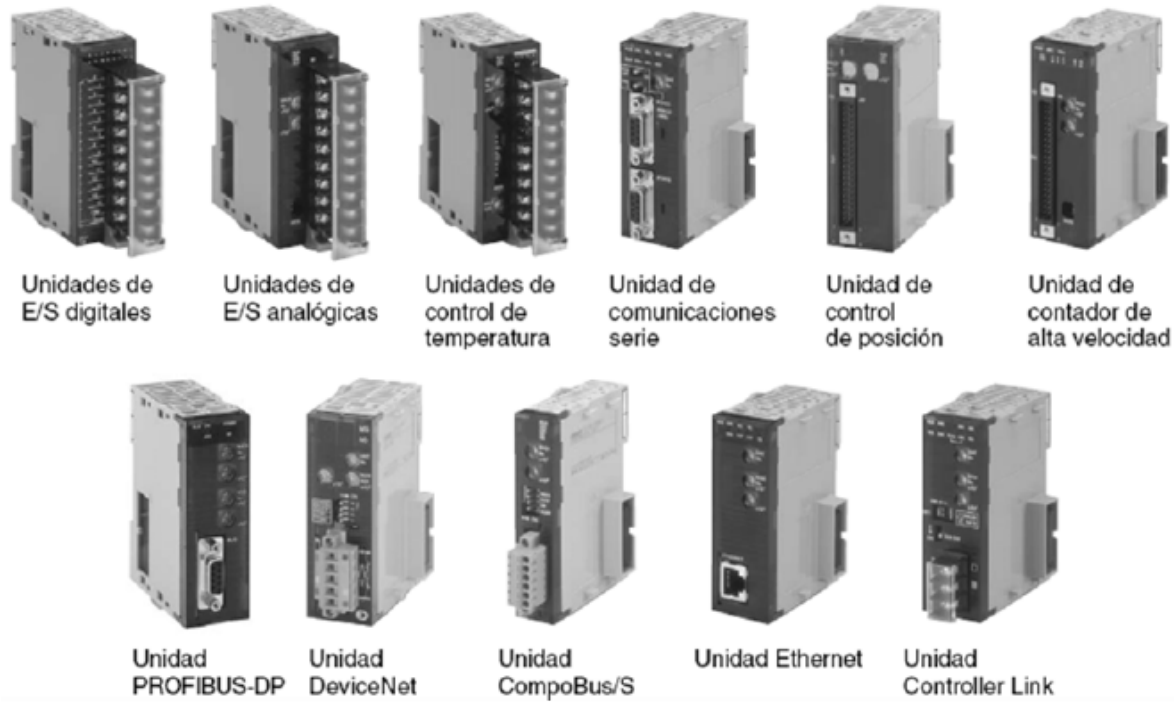


Fig. 6. Expansiones de del PLC de OMRON.

2.4.3. Registros de memoria del PLC

La clasificación de la memoria interna no se realiza atendiendo a sus características de lectura y escritura, sino por el tipo de variables que almacena y el número de bits que ocupa la variable. Así, la memoria interna del autómata queda clasificada en las siguientes áreas.

AREA DE DATOS		TAMAÑO	RANGO
CIO	Entradas /Salidas	1280 bits	CIO0000 – CIO0079
	Data link	3200 bits	CIO1000 – CIO1199
	Unidades de bus	6400 bits	CIO1500 – CIO1899
	Unidades especiales de entrada/salidas	15360 bits	CIO2000 – CIO2959
	PC Link serie	1440 bits	CIO3100 – CIO3189
	Entrada/Salidas integradas	10 bits + 6 bits (1 canal + 1 canal)	CIO2960 – CIO2961
	Device Net	9600 bits	CIO3200 – CIO3799
	Área interna de entradas/salidas	37504 bits/ 4800 bits	CIO1200 – CIO1499/ CIO3800 – CIO6143
AREA WR		8192 bits	W000 – W511
AREA HR		8192 bits	H000 – H511
AREA AR		15360 bits	A000 – A959
AREA TR		16 bits	TR0 – TR15
AREA DM		32768 bits	D0000 – D32767
AREA TEMPORIZADORES		4096 canales	T0000 – T4096
AREA CONTADORES		4096 canales	C0000 – C4096
AREA DE FLANCOS DE TARES		32 bits	TK00 – TK32
REGISTRO DE INDICE		16 registros	IR0 – IR15
REGISTRO DE DATOS		16 registros	DR0 – DR15

Tabla 3. Memoria del PLC.

La memoria del PLC la tenemos estructurada en dos áreas principalmente:

- Área de programa: Espacio donde queda almacenada el algoritmo.
- Área de datos: Espacio para escribir o leer datos del estado del PLC. Este se divide en 5 tipos de áreas de registros según la función que se necesite para el programa: IR, SR, AR, HR, LR, DM, TC.

Registros de E/S i Relés Internos (I/O and Internal Relay Area, IR).

Este espacio contiene las direcciones que se utilizan como entradas o salidas y, los relés auxiliares que facilitan la estructura del programa y es la más utilizada.

Esta área comprende dos tipos de canales: los asociados los terminales externos (canales E/S) y los canales internos, los cuales se llaman bits de trabajo. Estos últimos

no tienen control directo sobre los dispositivos externos y, son utilizados para controlar otros bits necesarios para dar estructura al programa.

Esta área contiene 228 canales de 16 bits (palabras) dividas en 4 partes:

- Canales de entrada (IR0 – IR9).
- Canales de Salida (IR10 – IR19).
- Área de trabajo (IR20 – IR49; IR200 – IR227).
- Área Reservada (el resto de bits).

Registros especiales (Special Relay Area, SR).

Esta área contiene relés para monitorizar operaciones del sistema, generar pulsos de reloj, señales de error, flags, bits de control y se encuentran entre SR228 hasta a SR255.

Registres Auxiliars (Auxiliar Area, AR).

Esta ocupa 24 canales para almacenar datos internos, pero algunos están reservados para funciones del sistema. El rango de canales útiles van des del AR0 hasta el AR23. Para el acceso a los bits se ha de utilizar el prefijo AR seguido del número de canal y del número de bit. El área AR mantiene los datos durante la caída de tensión.

Registros de Retención (Holding Relay Area, HR).

El área HR se utiliza para almacenar diversos tipos de datos. La característica principal de esta área, es que conserva el estado delante de fallos de alimentación o cambio de modo del PLC. Los canales de direccionamiento van des de HR0 hasta HR19. Para acceder los bits, el formato de dirección contiene el prefijo HR seguido del canal.

Registros de enlace (Link Relay Area, LR).

Contiene 16 canales que sirven para intercambiar datos entre PLCs unidos en red tipo PC LINK 1:1 (puerto serie). La parte del área que no se utiliza para la comunicación, puede ser utilizada para la manipulación y almacenamiento de datos internos de la misma manera que se utiliza el IR. El área LR no retiene la información, cuando falla la alimentación y, para el acceso a esta se necesita el prefijo LR seguido del número de canal y el número de bit.

Registros de memoria de datos (DM)

Sirve para la manipulación y almacenamiento de datos internos y, sólo se accede mediante canales de 16 bits. Por esta razón, no se pueden utilizar en instrucciones con operaciones del tipo bit. También mantiene los datos durante los fallos.

El rango de direccionamiento esta subdividida en cuatro partes:

• **Lectura/Escritura**: 2.026 palabras (DM0 – DM1999; DM2022 – DM2047).

• **Área de Error**: almacena códigos de errores. 22 canales (DM2000 – DM2021).

• **Solo Lectura**: No se puede sobre escribir des del programa. (DM6144–DM6599).

• **Configuración PC**: Elección de parámetros de Control. 56 canales. (DM6600-DM6655).

Temporizadores i contadores (Timer/Counter Area, TC).

Simula el funcionamiento de temporizadores y contadores, y se identifican mediante los 3 dígitos de canal (del TC000 al TC255) precedido de las siglas del elemento, TIM o CNT.

2.5. MODOS DE OPERACIÓN

Los modos de operación son la forma de trabajar que tiene la CPU del PLC y son modos de programa, modo monitor y modo run.

El modo programa sirve para realizar tareas previas a la ejecución del programa:

- Cambio de parámetros iníciales o de operación como la configuración de la CPU y/o sus unidades periféricas.
- Escritura, transferencia o revisión del programa.
- Forzado de los bits de E/S a set o a reset.

El modo monitor solo se usa para depurar el programa y realizar tareas del tipo:

- Edición del programa.
- Monitorizar áreas de memoria durante la operación.
- Forzar los bits de E/S a set o a reset.
- Para depurar.
- En nuestro caso, para que las simulaciones se pueden realizar se tiene que estar en este modo.

El modo run es cuando el programa se ejecuta en condiciones normales de tiempo de ciclo. Es el modo de operación en servicio normal del PLC. No se puede realizar acciones de cambio de parámetros o configuración, forzado de bits, cambiar PV (valor presente).

2.6. EJECUCIÓN DEL PROGRAMA EN OPERACIÓN CÍCLICA

La siguiente figura muestra la operación cíclica del CJ1M cuando el programa se ejecuta normalmente. Los resultados de la ejecución del programa son transferidos a la memoria de E/S inmediatamente después de la ejecución del programa (durante refresco d' E/S).

El tiempo de ciclo es la suma de los tiempos requeridos para la ejecución del programa, refresco de E/S y servicio de puertos periféricos.

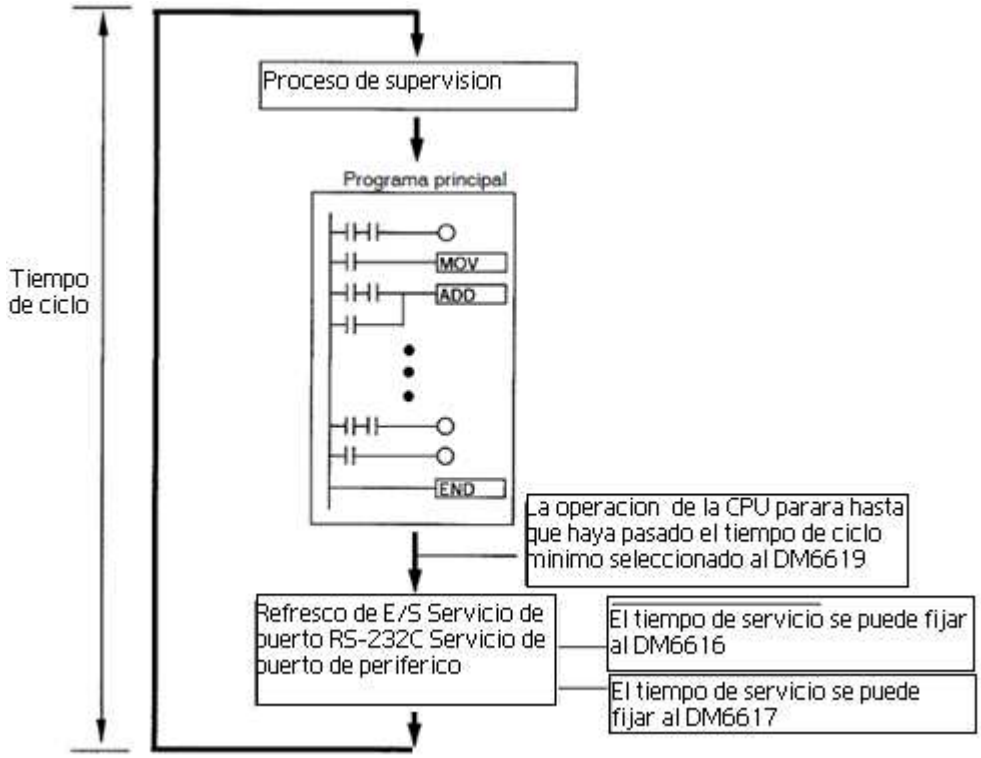


Fig. 7. Esquema de ejecución del PLC.

CAPITULO 3. DISEÑO DE LA PLANTA

En este capitulo ofrece información, de manera detallada, sobre los elementos a la hora de diseñar la planta. Es decir, una pequeña explicación del programa que uso para concebir las aplicaciones como también los elementos de comunicación para transmitir datos entre el PLC o simulador y la simulación. En este apartado hago una exposición más detallada sobre las librerías que uso para la transferencia de datos.

Para diseñar la simulación de la planta he usado el lenguaje de programación LABVIEW.

3.1. LABVIEW – PROGRAMA DE INTERFACE GRAFICA

LabVIEW significa Laboratory Virtual Engineering Workbench, que quiere decir laboratorio de ingeniería virtual de trabajo, y consiste en un lenguaje de programación gráfica para el diseño de sistema de adquisición de datos, instrumentación y control. Una de las características mas relevantes es la de poder crear fácilmente interfaces de usuario para la instrumentación virtual sin necesidad de elaborar código de programación. La manera de conseguirlo es mediante la construcción del programa con diagramas de bloques. Se basa en un modelo de programación de flujo de datos, que libera al programador de la rigidez de la arquitectura basada en texto.

Es un sistema de programación gráfica que tiene un compilador que genera código optimizado, donde la velocidad de ejecución es comparable al lenguaje C.

LabVIEW contiene una herramienta de asistencia capaz de detectar automáticamente cualquier instrumento conectado al ordenador, instalando los drivers apropiados y facilitando la comunicación con el instrumento.

En una fase inicial, LabVIEW fue creado para construir instrumentación virtual (osciloscopios, generadores de función, voltímetros, etc.). Pero en las últimas versiones se han utilizado ampliamente para desarrollar aplicaciones en el control de procesos, gracias a la amplia disponibilidad de tarjetas de adquisición de datos y a la facilidad de construir aplicaciones en un ambiente gráfico.

3.1.1. Estructura de programación

LabVIEW es un programa totalmente gráfico donde se utilizan bloques funcionales para conseguir el control y monitoreo necesario del proceso de fabricación, control, etc.

Inicialmente LabVIEW fue creado para construir instrumentación virtual, el archivo donde se genera el programa se llama Instrument Virtual o VI.(en ingles Virtual Instrument). Para poder hacer el programa, LabvieW ofrece dos plataformas de interface para cada VI: **el panel frontal y el bloque de diagrama.**

El panel frontal: es la interface que el usuario final verá, una vez el VI esté conectado en modo Run. Es donde se mostrará gráficamente el proceso a monitorizar, y los datos adquiridos que el instrumento virtual está midiendo. Lo más destacado es:

- Per crear el panel frontal se colocamos los controles e indicadores de la paleta de control
- Cada icono representa una sub-paleta, la cual contiene controles para colocar en el panel frontal.
- Un control es un objeto que utiliza el usuario para interactuar con el VI, introduciendo datos o controlando el proceso.
- Un indicador es un objeto del panel frontal que muestra datos al usuario.
- Cuando se coloca un control o indicador en el panel frontal, automáticamente aparece un terminal en el diagrama de blocs.

Fig. 8. Panel frontal de LabVIEW.

El diagrama de bloques: es la parte donde se hace visible para el usuario donde se crea el programa de flujo.

Vendría a ser como la parte de conexiones eléctricas del instrumento. Lo más destacado es:

- Para construir el diagrama de bloques se usan los terminales generados en el panel de control para los controles e indicadores, y los VIs, funciones y estructuras de la paleta de funciones.
- Cada icono de la paleta representa una sub-paleta, la cual contiene Vis y funciones para colocar en el diagrama de bloques.

- Las estructuras, VIs y funciones (llamados en conjunto nodos) de la paleta de funciones proporcionan la funcionalidad al VI.

- Cuando se colocan nodos a un diagrama de bloques, se pueden conectar entre sí y a los terminales generados por los controles e indicadores del panel de control mediante la herramienta de conexión (Wiring Tool) de la paleta de herramientas.

- Al final, un diagrama de bloques completo se parece a un diagrama de flujo.

Una de les características es la descomposición modular, que permite crear Sub-VI dentro de un VI principal. Sería equivalente a subrutinas en programación estructurada.

Es un sistema muy flexible y abierto, porque cualquier fabricante de instrumentos o de tarjetas de adquisición de datos puede proporcionar el controlador de su producto mediante un VI, dentro del entorno de LabVIEW.

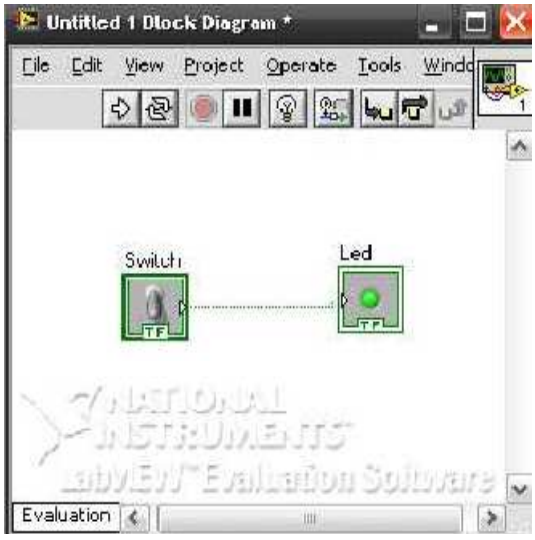


Fig. 9. Diagrama de bloques de LabVIEW.

3.1.2. Ejecución de un VI

Una vez se ha acabado la programación del VI, se procede a la ejecución. Nos hemos de situar a la parte de arriba y pulsar el botón de Run, situado a la barra de herramientas.

Fig. 10. Símbolo de LabVIEW de ejecutar.

El programa comenzará a ejecutar. Mientras dura la ejecución del mismo, la apariencia del botón de Run es la que se muestra a continuación:

Fig. 11. Indicadores de ejecución de LabVIEW.

De esta manera el programa se realizará una sola vez. Si se desea una ejecución continua, se pulsa el botón situado a la derecha del de Run (Continuous Run). Si durante el funcionamiento continuo del programa se vuelve a pulsar este botón, finalizará la última ejecución del mismo, después del cual el programa se parará.

Para finalizar la ejecución de un programa se puede operar de dos maneras. La primera, y la más aconsejable, es pulsar un botón en el panel frontal del VI, la pulsación produce la interrupción del bucle de ejecución de la aplicación. La segunda forma para parar la ejecución del VI es pulsando el botó de pausa o el de stop. La diferencia de ambos es que si se pulsa stop, la ejecución del programa finaliza inmediatamente, mientras que si es pulsa pausa, se producirá una detención en el funcionamiento del programa, retornando a la ejecución una vez se vuelve a pulsar el mismo botón.

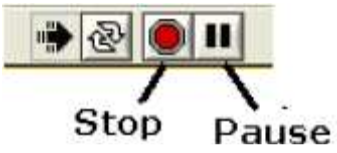


Fig. 12. Indicadores de ejecución de LabVIEW.

3.2. CX-SERVER

CX-Server es un sistema de gestión de comunicaciones de Microsoft Windows para PLC de OMRON. Dichas librerías facilitan la comunicación de los dispositivos que se quieran comunicar con los autómatas de OMRON. En nuestro caso se encarga de gestionar la comunicación entre el PLC y nuestras prácticas.

CX-Server consta de los siguientes componentes:

Además de actuar como un servidor para el software de OMRON otros, CX-Server incluye herramientas de otros usuarios:

- **CX-Server DDE Administrador**. Es una aplicación que permite la transferencia de datos entre un PLCs y un cliente que utiliza DDE, por ejemplo Microsoft Excel.

- **CX-Server Importación.** Esta herramienta se utiliza para las direcciones y la configuración del PLC se define utilizando productos CVSS y LSS. Permite la definición de direcciones y la configuración del PLC para ser importados a proyectos del CX-Server.

- **CX-Server monitoriza el rendimiento**. Esta herramienta sirve para monitorear las comunicaciones y mostrar el nivel de rendimiento actual del CX-Server (es decir, un chequeo para ver si CX-Server se sobrecargar).

- **CX-Server Administrador de controladores**. Es una herramienta para la instalación y desinstalación de nuevos controladores para dispositivos que queramos controlar bajo la aplicación CX-Server.

CX-Server incluye varios ejecutables, librerías de enlace dinámico (DLL) y sus componentes.

Para utilizar ciertos tipos de comunicación (Ethernet, es decir, SYSMAC LINK, SYSMAC NET), algunas configuraciones se debe configurar antes de su uso antes de la instalación del software.

Esta información se mantiene en un archivo de proyecto CX-Server con una extensión. CDM. Este archivo contiene toda la información sobre el PLC, que CX-Server puede conectarse, y las direcciones de interés en cada PLC que acceda.

Cada archivo del proyecto CX-Server es independiente y es similar en concepto a un documento. CX-Server puede tratar con muchos archivos de proyecto de servidor a la vez, aunque a menudo sólo un proyecto CX-Server es utilizado por el software cliente en cualquier momento.

La serie de pasos que están involucrados en la creación de un proyecto CX-Server, en un alto nivel, que pueden considerar como:

- La identificación de los autómatas que el PC es comunicarse con;

- La identificación de las direcciones (puntos) en el PLC que se puede acceder durante comunicaciones;

- Establecer el tipo de red que se utilizará.

CX-Server proporciona:

- Soporte global para los PLCs de la serie C, CJ, CV y CS.

- Soporte para Sysmac Way, Sysmac Net, Sysmac Link,

- Controller Link, Ethernet y ToolBus.

- Posibilidad de soportar los nuevos PLCs que aparezcan simplemente actualizando a la última versión de CX-Server.

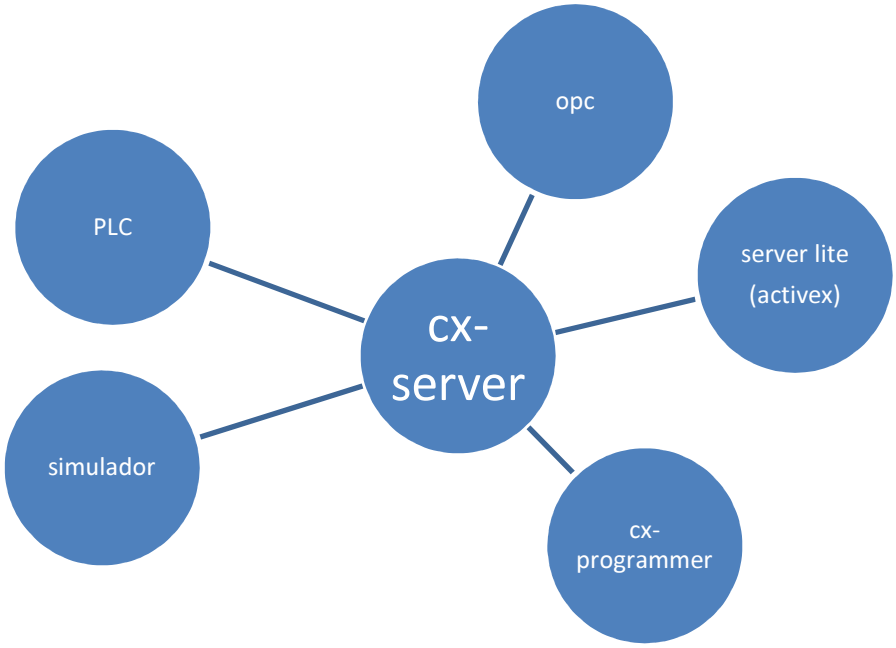


Fig. 13.Esquema de comunicación del CX-SERVER con las demás aplicaciones y periféricos.

3.2.1. CX-Server Lite

Directamente no se puede comunicar las aplicaciones con el CX-Server si no que necesitamos una interface. En nuestro caso, dicha interface es el CX-Server Lite, el cual funciona bajo la tecnología ActiveX.

El CX-Server Lite permite que los datos del PLC sean recogidos por el software de comunicaciones del CX-Server de OMRON el cual trabaja con el cliente, entre ellos el programa LabVIEW. Y permite que los datos del proceso existentes sean adquiridos y analizados. Además, incluye algunos componentes gráficos permitiendo una recreación fácil en los MMI. El MMI es el interfaz de unión entre el operario y la máquina. Puede ser un panel de operador o una computadora (PC), pero en ambos casos comunican y transmiten datos a y desde el PLC.

El software CX-Server Lite de OMRON le garantiza una adaptabilidad total. Permite que el software ofimático estándar como Word, Excel, Visual Basic, VBA y Delphi y por supuesto LabVIEW acceda a los dispositivos OMRON de automatización usando la norma ActiveX. CX-Server Lite contiene una colección concisa de componentes gráficos especializados que permiten una construcción fácil de aplicaciones. Las conexiones con los paquetes de software se crean, cambian o añaden “arrastrando y soltando” componentes gráficos y rellenando cuadros estandarizados. CX-Server Lite es una versión completa de CX-Server, pero accediendo solamente mediante controles ActiveX.

Toda la funcionalidad de comunicaciones es gestionada por el CX-Server.

Los componentes ActiveX sirven para comunicarse entre sí, el PLC y el cliente, con la finalidad de usar el código de los demás. Los objetos no necesitan conocer por anticipado en qué objetos se van a comunicar, ni su código necesita estar escrito en el mismo lenguaje.

CX-Server control de comunicación

Este control proporciona un interfaz integra entre el programa para las aplicaciones del cliente del CX-Server Lite como LabVIEW, Excel, Visual Basic y la comunicación de OMRON, CX-Server. Observe que el control solamente es visible cuando el uso del cliente está en el modo del diseño.

Fig. 14. Icono del CX-Server en el panel frontal de LabVIEW.

ActiveX

ActiveX es el nombre que recibe la tecnología que permite reutilizar un código con una interface definida. De este modo se puede llegar a hacer que el código sea accesible por otras aplicaciones y así ir enlazando unas aplicaciones con otras.

Por tanto, ActiveX nos da un entorno en el que mediante unos servicios basados en objetos se permite a diversos componentes comunicarse entre sí para reutilizar el código de los demás, y así poder enlazar unos programas con otros.

Una aplicación es compatible con la automatización de un proceso ya sea como servidor o como cliente. Las aplicaciones que proporcionan los objetos y métodos de funcionamiento de los objetos ActiveX son servidores de automatización. Las aplicaciones que utilizan los métodos expuestos por otra aplicación son los clientes de automatización de ActiveX. En conclusión, LabVIEW puede funcionar como ActiveX servidor o como ActiveX Cliente.

El ActiveX de la compañía OMRON sirve para comunicar LabVIEW con los dispositivos de OMRON, es decir tiene las librerías estándar pertinentes de comunicación de OMRON.

Para usar el CX-Server Lite en LabVIEW hay que usar los controles y funciones ActiveX. Hay una paleta donde encontramos el contenedor ActiveX.

3.3. USO DE CX-SERVER LITE EN LABVIEW

3.3.1. Librerías de comunicación

Como hemos dicho antes, la programación de LabVIEW esta orientada para programar por objetos. Estas librerías o funciones tienen propiedades y métodos que se pueden modificar (escribir) u obtener información de ellas (leer).

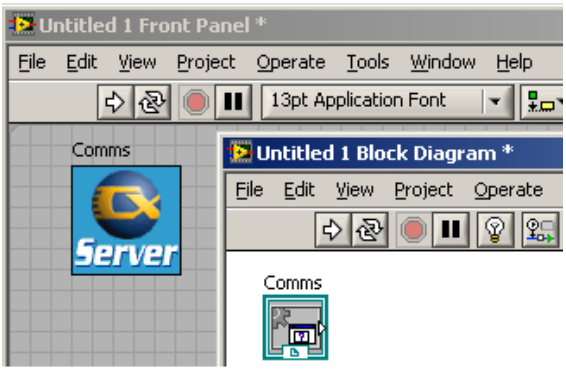


Fig. 15. Icono de comunicación del CX-ServerLite en el diagrama de contactos de LabVIEW.

Clicando con el botón derecho del ratón sobre el icono del ActiveX de comunicación del diagrama de bloques y se nos aparecerá un cuadro de dialogo en cascada donde seleccionaremos los métodos que vamos a emplear para la comunicación.

Al principio de nuestro proyecto hemos empleado las siguientes librerías:

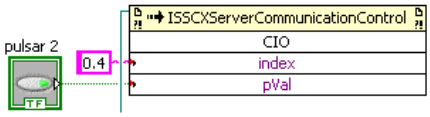


Fig. 16. Librería de escritura en el área CIO.

En esta función le indicamos:

- El espacio de **index** de memoria es un punto donde guardamos una variable binaria en la posición 0.4.

- En **pVal** ubicamos la entrada que queremos guardar en la posición 0.4

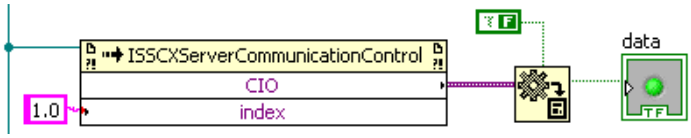


Fig. 17. Lectura de un bit en la memoria del PLC del espacio CIO.

En esta función le indicamos:

- El espacio de **index** de memoria es un punto donde leemos la variable binaria que está en la posición 0.1.

Con este método de comunicación me surgían dos problemas:

Uno era que tenía que colocar tantas instrucciones de lectura como de escritura para poder controlar todas las variables que gobiernan la planta, por lo tanto el tiempo de lectura y escritura se hacían extremadamente largos por que en cada iteración global tenía que llamar a todas las funciones de comunicación.
Y el segundo problema que se me presento fue que en cada iteración global llamaba a las funciones de escritura del espacio de memoria CIO sin que las entradas hubieran sido modificadas.

3.3.2. Evolución de la comunicación de las librerías con métodos de LabVIEW

A continuación expondremos las soluciones adoptadas:

a) Para solucionar el primer problema en el cual trabajamos con bits, barajamos la idea de trabajar con registros. Es decir que en cada iteración global solamente hacemos una lectura y una escritura en los registros del espacio de memoria CIO del PLC. Con ello reducimos considerablemente los tiempos de comunicación.

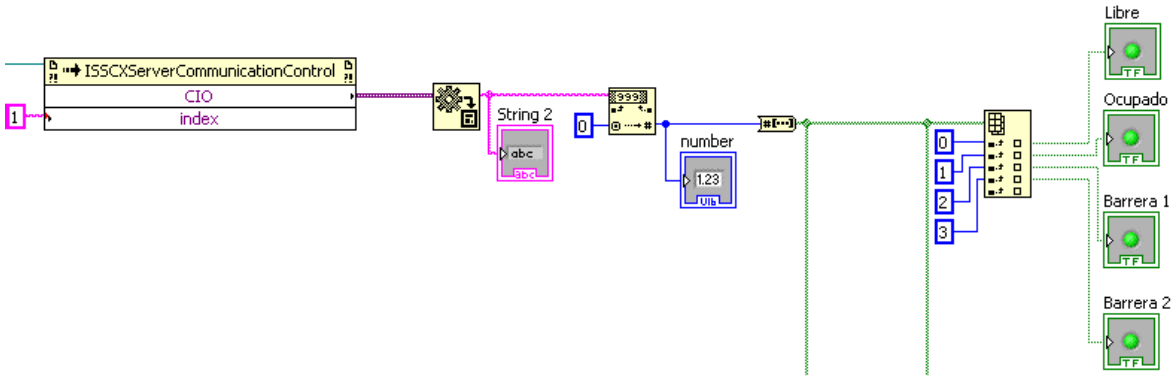


Fig. 18. Lectura del registro de salida 1 del espacio de memoria CIO del PLC.

Como se puede ver en la imagen superior lo que hacemos es leer toda una palabra y luego la desmenuzamos para obtener los estados de cada bit de la palabra en cuestión.

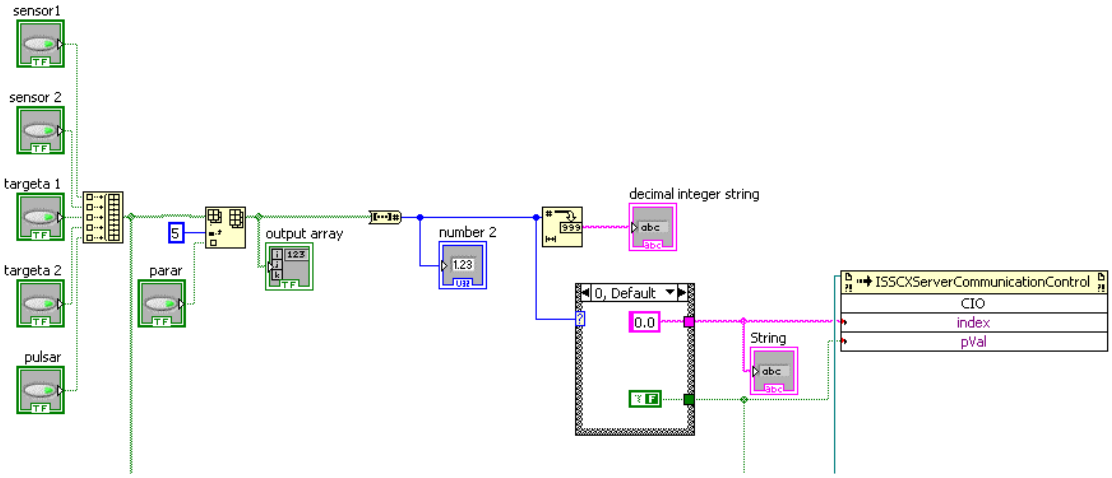


Fig. 19. Escritura del registro 0 de entrada del espacio de memoria CIO del PLC.

La idea es colocar los estados de las variables de entrada en un vector unidimensional, y ordenarlos según como lo tengamos configurado en el espacio de memoria de entrada CIO. Dicho vector, los cuales son estados binarios lo pasamos a decimal, es decir cada vector escrito tiene un numero asociado.

Cuando una de las entradas se activa, se le asocia el número que está en la estructura "case". Dicha estructura, está diseñada para cuando se active una de las entradas se le asocie al bit de entrada del espacio de memoria CIO, y la información booleana será verdadera.

b) Para solucionar este segundo problema que me surgió, hay que decir, que solo sirve para las instrucciones de escritura, ya que son estas las que gobierna la planta, en cambio para las instrucciones de lectura es producto de los cambios en las entradas.

El propósito de esta modificación es ahorrarnos tiempo, cuando no interesa escribir por que no habido cambios en las entradas de la planta.

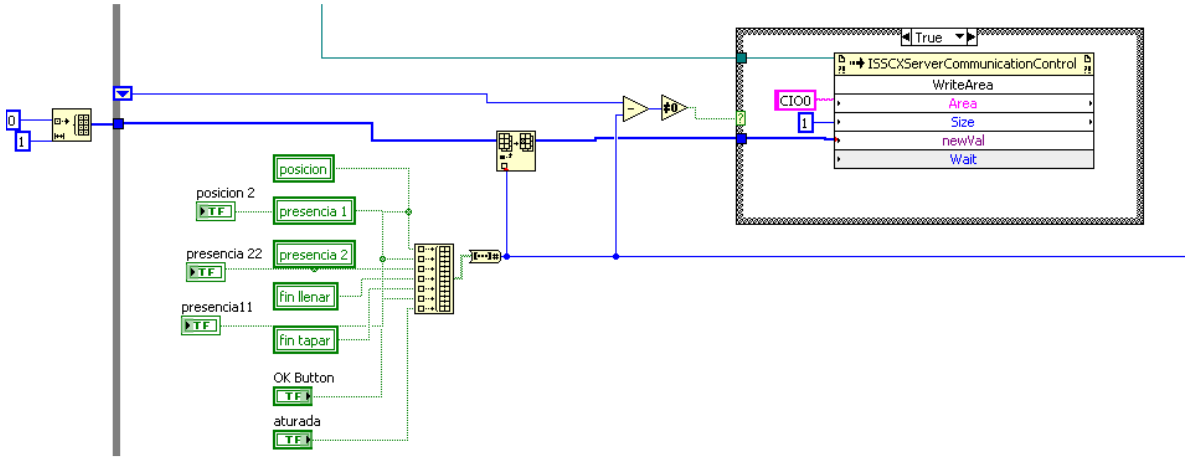


Fig. 20. Escritura del registro 0 de entrada del espacio de memoria CIO de PLC.

En este caso lo que hacemos es escribir en el espacio de memoria cuando hay un cambio en las entradas de la planta. La idea es que si no hay cambios en la entrada entonces la función de escritura no se habilita.

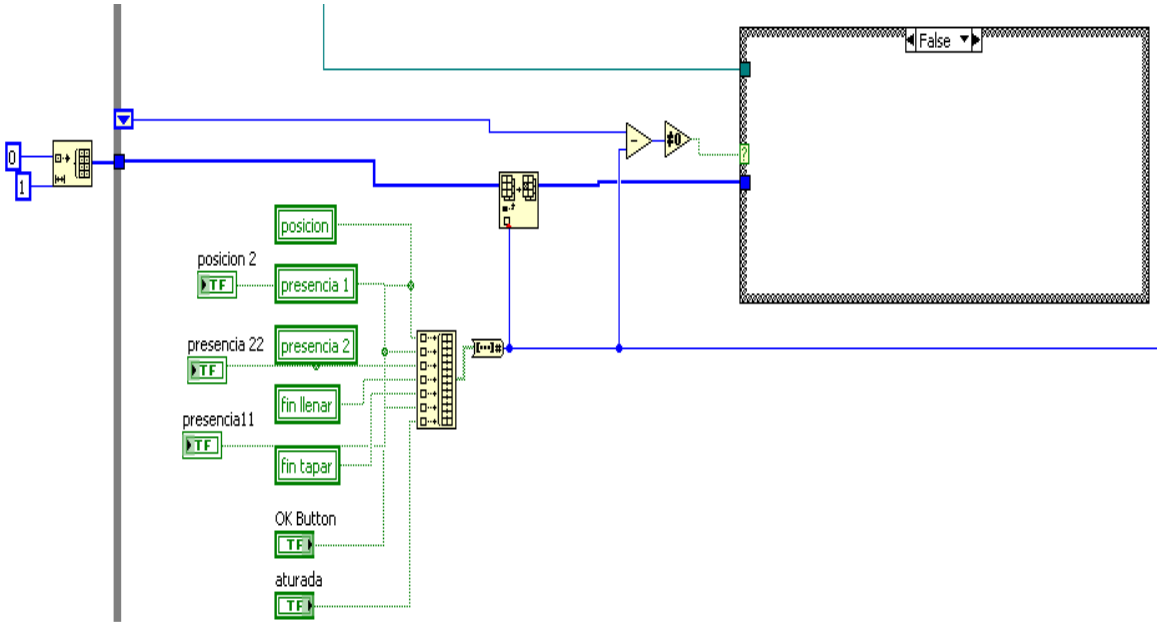


Fig. 21. Cuando las variables de entrada no han sido modificadas en el siguiente ciclo.

En este caso como no hay cambios en las entradas del sistema, es decir los estados anteriores son los mismos que los actuales y por lo tanto la estructura “case” es falsa y deshabilita la función de escritura.

También hay que tener en cuenta que la función de escritura es diferente.

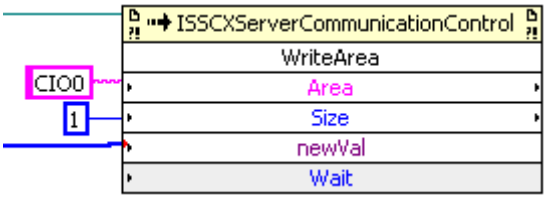


Fig. 22. Función de escritura WRITE AREA.

En esta función le indicamos:

- El espacio de **Area** de memoria en nuestro caso es CIO0.
- El número de registros a escribir **size_** es 1.

- Y por ultimo en **newVal** ubicamos la información de los estados de las entradas. En este punto hay que reseñar que los datos de AREA que se manejan los PLCs de OMRON son de 16 bits por lo tanto en este apartado la información tiene que estar definida en 16 bits.

3.3.3. Optimización de la comunicación

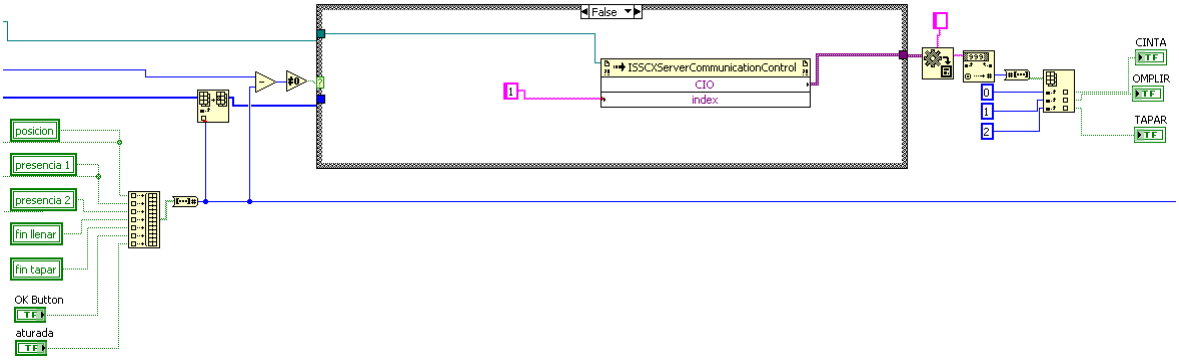


Fig. 23. Optimización de programación de comunicación con las librerías de LabVIEW con cambio de estado en las entradas.

Como se puede ver en la imagen de arriba esta es la solución definitiva para la comunicación. Es una versión mejorada del apartado anterior. En este caso tenemos las funciones de escritura y de lectura en el mismo caso.

Su funcionamiento es el siguiente cuando los estados de las entradas se modifican entonces entramos en la estructura “case”. En dicha caso enviamos los estados de las entradas a sus respectivas ubicaciones del espacio de memoria CIO luego dejamos cierto tiempo para que el PLC las procese, y cuando ya ha pasado dicho tiempo leemos las salidas, es decir leemos el registro donde tenemos emplazado en el espacio de memoria CIO del PLC.

Cuando los estados de las entradas no se modifican, cuando los estados de las entradas son las mismas que los estados anteriores, no hace falta escribirlos y solo queremos leer las salidas, por lo tanto el otro caso de la estructura “case” será:

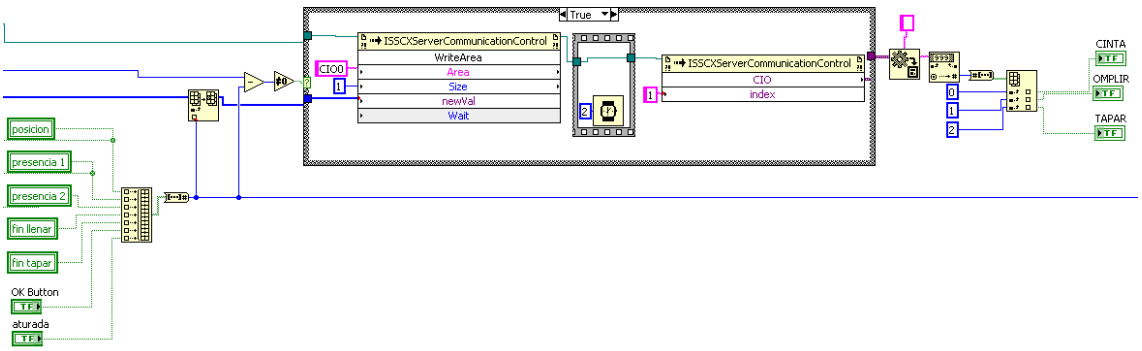


Fig. 24. Optimización de programación de comunicación con las librerías de LabVIEW, sin cambio de estado en las entradas.

El cual solo leerá el registro de las salidas del espacio de memoria CIO que ha procesado el PLC.

Por ultimo, explicare una librería de comunicación la cual esta diseñada para leer el espacio de memoria de los PLCs OMRON donde se almacena la información de los contadores:

Fig. 25. Función de lectura dela área de contadores del autómata.

En esta función le indicamos:

- En el sector de **index** le indicamos en que sección del espacio de memoria de los contadores tenemos la información que queremos. (Esta función la empleamos en la práctica del parking).

3.3.4. Creación de un proyecto

Si queremos usar el CX-Server Lite necesitamos crear un proyecto en la aplicación cliente el cual definiremos los parámetros del PLC a utilizar como el tipo red que emplearemos en la comunicación.

Para insertar controles o indicadores en el panel frontal como insertar funciones, operadores, etc. Usamos la paleta, dicha paleta está organizada por tipos. En nuestro caso queremos insertar el contenedor ActiveX

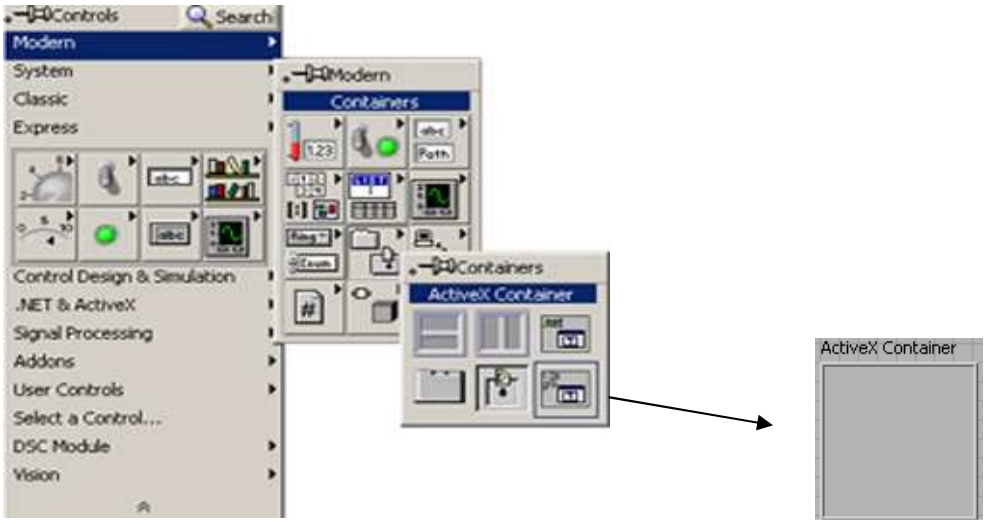


Fig. 26. Ruta donde podemos encontrar el Activex

Una vez que ya tenemos colocado el contenedor de ActiveX en el panel frontal, clicamos con el botón derecho sobre el objeto y elegimos “insert ActveX Object”

Se nos abrirá una ventana donde nos saldrá todos los ActiveX que tengamos en el ordenador.

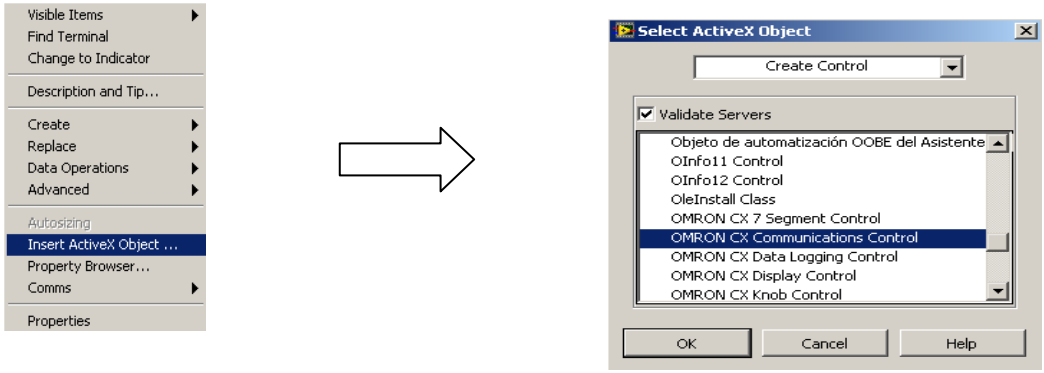


Fig. 27. Ventanas donde se encuetra el ActiveX de comunicación.

Nosotros solo tenemos que buscar el ActiveX de comunicación del cliente de OMRON.

Fig. 28. Icono del Gestor de comunicaciones de OMRON

Y nos saldrá el icono del ActiveX de comunicación de OMRON.

Ya tenemos configurado el cx-server lite para labview. El siguiente paso será crear un proyecto para asignar el PLC que queremos que gobierne nuestras aplicaciones y la configuración que gestione la comunicación.

Crear proyecto

El primer paso antes de empezar tenemos que crear un proyecto con extensión. CDM, si no lo tenemos antes.

Ahora clicamos con el botón derecho sobre el icono de comunicación para configurar los parámetros de la comunicación y tipo de autómata a usar.

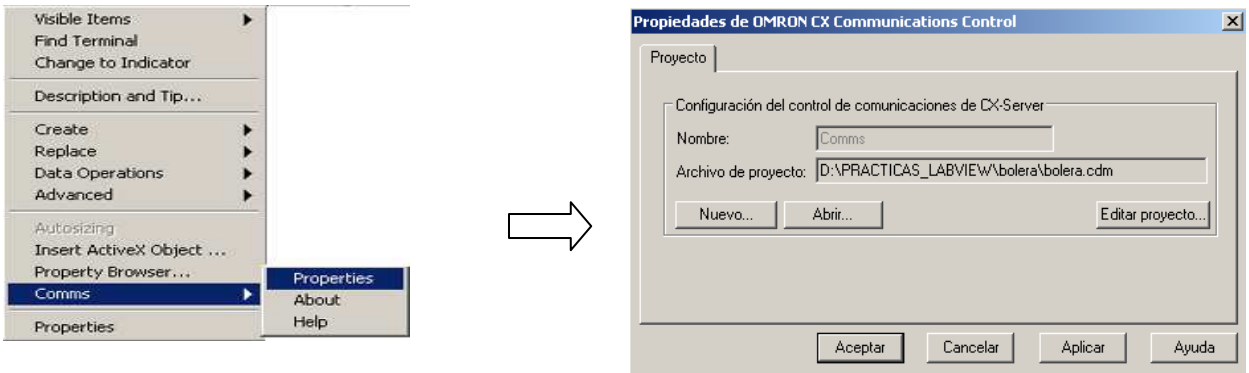
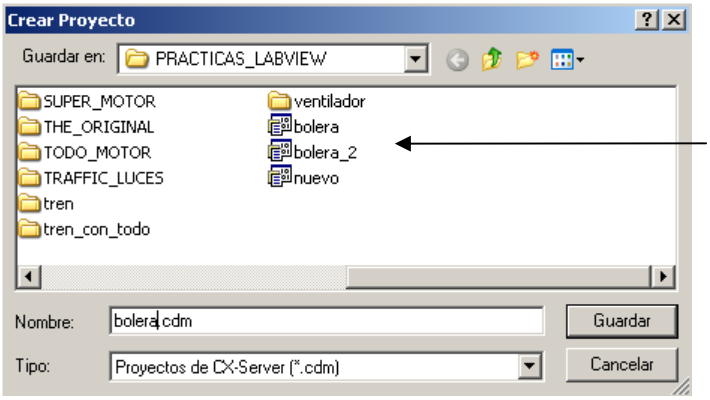


Fig. 29. Ventanas para crear un proyecto de comunicación en LabVIEW.

Nos saldrá una ventana en cascada como se ve en la figura. Entonces seleccionamos en el apartado “comms” y clicamos en “properties”.

En esta ventana creamos el proyecto donde vamos a configurar los puntos de entrada y de salida de la planta y el tipo de plc que lo va a gestionar.

Clicamos en “Nuevo” para crear un nuevo proyecto. En esta ventana creamos un proyecto donde le damos nombre y ubicación en la raíz donde los vamos a guardar.



. El archivo del proyecto tiene la extensión .CDM
.El nombre del dispositivo es independiente del nombre del proyecto. Se puede cambiar el nombre del proyecto o bien el nombre del dispositivo

Fig. 30. Ventanas ventana donde guardamos el proyecto.

Después clicamos en “Editar proyecto” y nos aparecerá una ventana donde configuraremos el tipo de PLC

Cuando ya hemos formado el proyecto, entonces nos hace falta configurar los parámetros del PLC que vamos a emplear para nuestras prácticas. Para ello tenemos que editar un nuevo PLC en el editor de proyectos.

Editar PLC

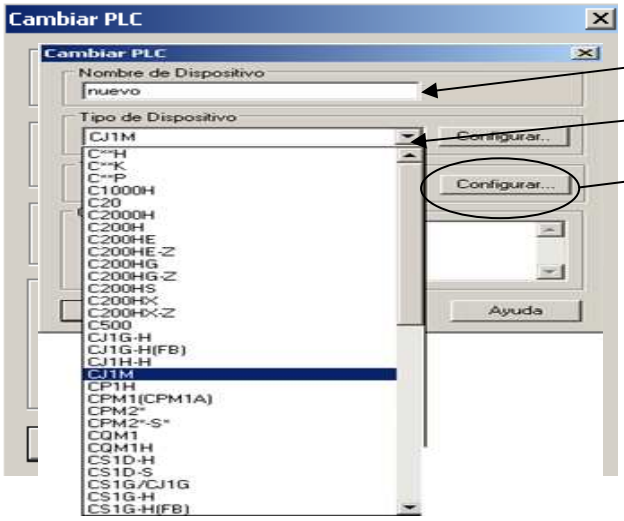
En este apartado seleccionamos el tipo de PLC, es decir, escogemos la CPU del autómata

Nombre del proyecto

Clicamos en el botón de “añadir PLC” para editar el tipo de PLC

Fig. 31. Ventanas donde editamos el proyecto.

Esta ventana tiene dos funciones, una para elegir el tipo de PLC que vayamos a utilizar y también el modelo de CPU que gestionara como también el tipo de red de comunicación que usaremos para conectarnos con el PC.



1- Le damos nombre al dispositivo
2- Seleccionamos el tipo de PLC
3- Haga clic en configurar y se nos abrirá las siguientes ventanas

Fig. 32. Ventanas donde configuramos el PLC.

4- Aquí escogemos el tipo de Procesador para nuestro PLC

5- Y por ultimo elegimos la capacidad de memoria

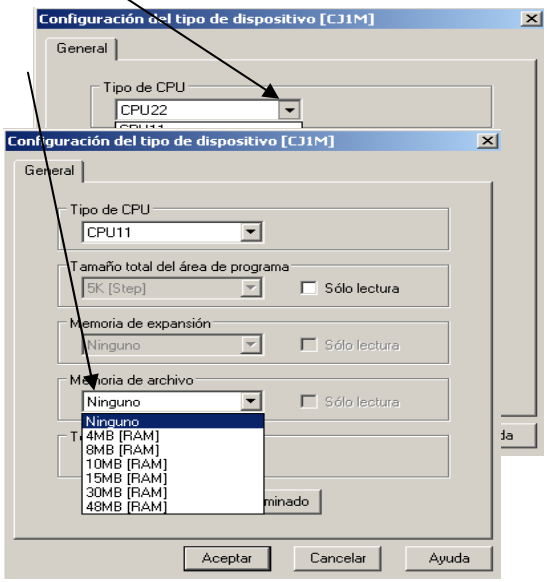


Fig. 33. Ventanas donde encontramos los parámetros de configuración.

Nota: En el último apartado, cuando escogemos la capacidad de memoria, se puede obviar en el caso que usemos el simulador CX-Simulator de OMRON.

Cuando ya hemos elegido el PLC, y configurado todas sus características necesarias para funcionar, entonces tenemos que modelas los parámetros de comunicación de nuestro PLC con la planta a controlar, o en nuestro caso, con la simulación que tenemos en nuestro terminal.

Configuración de red

En este apartado configuraremos la comunicación que asistiremos entre el PC y el PLC.

Como se puede ver poseemos diversas posibilidades para comunicarnos:

- **Control link**: Se utiliza cuando está montada en el ordenador una Tarjeta de Soporte de Controller Link para una -conexión directa con el PLC.

- **Ethernet:** Opera de manera similar a SYSMAC WAY excepto que transmite datos en formato binario.

La conexión al PLC se realiza a través del puerto de periféricos mediante el protocolo UDP.

Para el ordenador y el PLC, consiste de un número de red y número de nodo. Esto se introduce en el apartado.

Para poder conectar un PLC en una red Ethernet se necesita saber su dirección IP. La dirección IP, -necesaria Driver del diálogo Configuración de Red.

- **Ethernet (FINS/TCP):** Es lo mismo que Ethernet pero usa el protocolo TCP

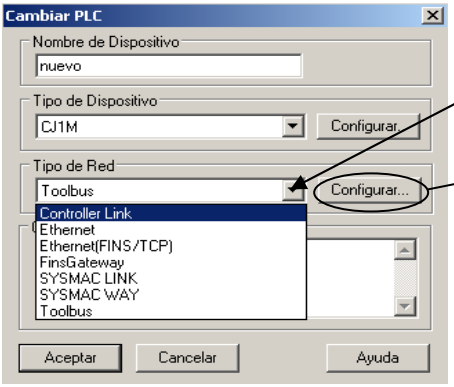
- **FinsGateway:**Es un programa de comunicación que proporciona OMRON

- **SYSMACLINK**: es una red de comunicación industrial de alta velocidad (2Mbits/s) que soporta la -conexión de múltiples PLCs. Se necesita una tarjeta de ordenador para SYSMAC LINK.

- **SYSMACWAY:** SYSMAC Way es un protocolo en serie desarrollado por Omron Electronics proporciona servicios de comunicación para C de Omron de la serie de PLCs

- **Toolbus:** Opera de manera similar a SYSMAC WAY excepto que transmite datos en formato binario.

La conexión al PLC se realiza a través del puerto de periféricos.



1- Clicamos en la pestaña para seleccionar el tipos de comunicación en nuestro caso será “control link”

2- Haga clic en configurar y se abrirá otra ventana en la cual definiremos los parámetros de comunicación

Fig 34. Ventana donde encontramos los parámetros de configuración.

3- En **“Direccion de destino FINS”** solo tenemos que modificar el Nodo por que usamos un simulador de PLC

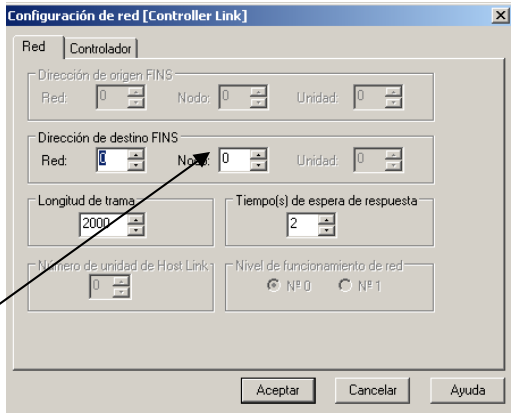
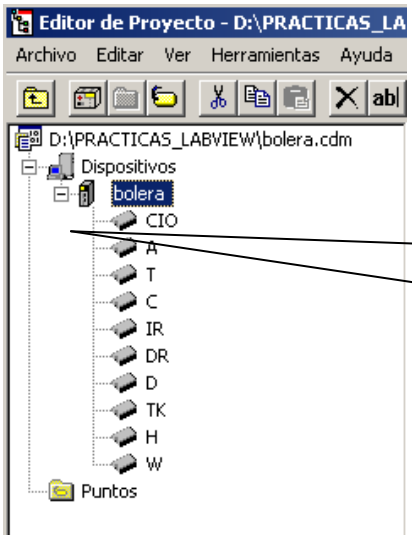


Fig 35. Ventana donde de configuracio de comunicación.

Nota: Los siguientes parámetros se pueden obviar cuando el PLC es un simulador ya que los tiempos de espera y las tramas las gestiona el simulador.

Crear puntos

Los puntos de comunicación sirven para vincular los distintos elementos del dibujo, y los botones a las marcas, entradas y salidas del PLC mediante el menú de puntos del sistema. Los puntos del sistema son los pasos que queremos visualizar del proceso.



Organizador del dispositivo- vista general grafica del proyecto. Use el organizador para desplazarse por los componentes del proyecto.

Para abrir los diferentes departamentos solo hay que clicar sobre el signo +.
Para cerrar los diferentes departamentos solo hay que clicar sobre el signo-.

Fig. 36. Ventanas del organizador del dispositivo

Los valores de los puntos de comunicación se guardan en unas áreas de memorias E/S del PLC.

Esta región de la memoria, contiene las áreas de datos a las que se pueden acceder mediante los operadores de las instrucciones. Estas áreas de datos incluyen:

CIO: Área de entrada y salida se usa para intercambio de datos.

A: Área de trabajo sin retención. Han sido diseñadas como indicadores y controles para operaciones de supervisión y control.

T: Área de trabajo con retención, hay 4096 temporizadores.

C: Área contador, hay 4096 contadores.

IR: Registro de índice. se utilizan para el direccionamiento indirecto.

DR: Registro de datos. Se usa para desplazar las direcciones de memoria del PLC en los registros de índice al direccionar los canales indirectamente.

D: Área de memoria de datos, se usa para el almacenamiento y manipulación de datos.

TK: Área de indicador de tareas.

H: Área de retención, estas áreas solo se pueden usar en el programa.

W: Área de trabajo no retentiva.

Definir punto

Definimos los puntos para identificar el formato de los datos llevados a cabo en la dirección del PLC. Esto permite que el CX-Server convierta automáticamente las variables del PLC en el formato específico de los datos a un formato común que la PC puede utilizar.

Para especificar físicamente los puntos entre el PLC y el PC, CX-SERVER tiene el “Editor de Punto”. En esta ventana tendremos que llenar 5 campos, entre ellos tendremos que precisar el formato de la variable.

Bit : bit

Signed Character Binary: Carácter binario con signo.

Unsigned Character Binary: Carácter binario sin signo.

Raw Character Binary: Carácter binario sin definición.

Single Word Unsigned Binary: variable de una cadena binaria sin signo.

Double Word Unsigned Binary: variable de dos cadenas binarias sin signo.

Quad Word Unsigned Binary: variable de cuatro cadenas binarias sin signo.

Single Word Signed Binary: variable de una cadena binaria con signo.

Double Word Signed Binary: variable de dos cadenas binarias con signo.

Quad Word Signed Binary: variable de cuatro cadenas binarias con signo.

Single Word Unsigned BCD: variable de una palabra sin signo en formato BCD.

Double Word Unsigned BCD: variable 2 palabras sin signo en formato BCD.

Quad Word Unsigned BCD: variable 4 palabras sin signo en formato BCD.

Single Word Signed BCD: variable de una palabra con signo en formato BCD.

Double Word Signed BCD: variable dos palabras con signo en formato BCD.

Quad Word Signed BCD: variable de 4 palabras con signo en formato BCD.

Double Word Float: variable de dos palabras en formato de coma flotante.

IEEE Float: variable de dos palabras para definir la coma flotante.

Complex: variable que define números complejos.

LReal: variable de 4 palabras (64Bit) con formato en coma flotante.

Ahora veremos como se definen los puntos que nos asocian las variables del sistema a controlar en el ActiveX de comunicación de OMRON de LabvieW.

1. Clicamos en “añadir punto” 

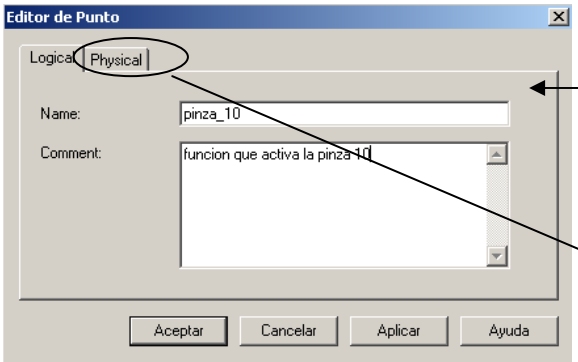


Fig. 37. Ventanas donde definimos el punto.

4. Elegimos el proyecto que hemos creado

5.Escribimos la dirección del punto de comunicación.

6. Escogemos el número de elementos a partir de la dirección.

7. Elegimos el tipo de dato.

8.Este ultimo apartado es opcional, aquí definiremos el estado de la variables.

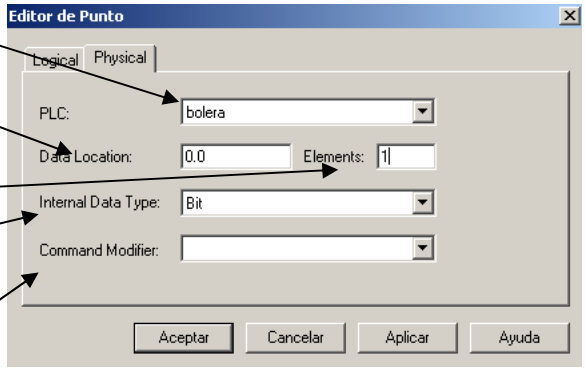


Fig. 38. Ventanas donde definimos el punto.

CAPITULO 4. Diseño de las aplicaciones

En el siguiente capitulo expondremos una pequeña exposición del funcionamiento de cada práctica como también sus inconvenientes en la programación. También explico los elementos mas utilizados en el panel frontal.

A la hora de diseñar las plantas de automatización, me encontré con la dificultad de que en el panel frontal de LabVIEW no tenia los suficientes elementos para caracterizar una buena simulación. A causa de este problema, tuve que buscar la manera de adaptar los elementos del panel frontal para lograr los objetivos.

Para simular el movimiento del proceso en algunas planta, empleo un contenedor de imágenes que lo configuro de tal forma que parezca un sistema real.

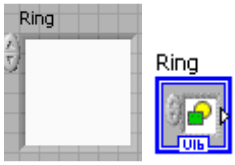


Fig. 39. Contenedor de imágenes y su terminal en el diagrama de bloques.

Para recrear movimientos rectos utilizo un deslizador que se puede modificar para insertar imágenes en el curso.

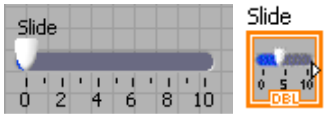


Fig. 40. Deslizador del panel frontal y su indicador en el diagrama de bloques.

4.1. APLICACIÓN 1. PARKING

En esta aplicación vamos a simular la entrada de un parking, la cual esta estructurada con una entrada y una salida.

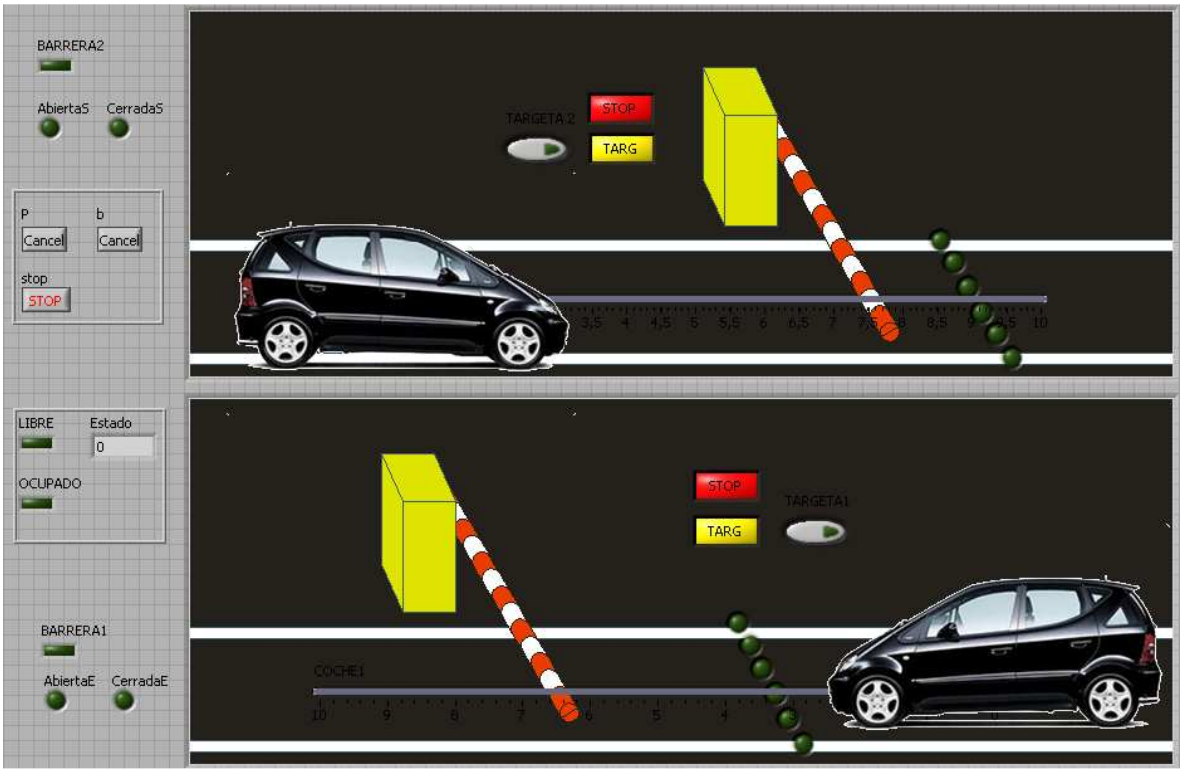


Fig. 41. Panel frontal de la práctica del parking.

4.1.1. Funcionamiento

Una vez que el automatismo se ha puesto en marcha, a través del pulsador P, su funcionamiento es el siguiente:

Al principio, el señal de libre esta encendido y el contador a cero. Entonces, si por ejemplo queremos entrar al parking tenemos que pulsar el pulsador de TARG y, si el parking está libre la barrera se abrirá y el contador se incrementara una unidad, y cuando el coche pasa por la célula 1, un temporizador empieza a contar hasta 3 segundos, pasado este tiempo la barrera se bajará. Pero en el caso que el coche no haya pasado por el sensor, la barrera se cerrará en 5 segundos.

Otro caso posible, es si seguidamente después de pasar el primer coche pasa otro coche detrás de él cuando la barrera esta abierta, y por lo tanto, lo detecta el sensor. En este caso, el contador se incrementa otra unidad, y el contador cuenta hasta 3 segundos antes de cerrar la barrera.

Y por último, puede ser posible que un coche entre cuando el parking está lleno, en este caso, el contador tiene que incrementar una unidad, por consiguiente, cuenta los coches reales que están dentro del parking, y la barrera no se cerrara antes de los 3 segundos después que de que haya pasado por el sensor.

No se considera que el coche vaya hacia atrás.

En el caso, que el coche quiera salir tiene que hacer el mismo procedimiento que en el caso de salida pero la célula 2 está antes que la barrera.

Cuando el contador llega a la cantidad de vehículos que puede contener el parking (4 coches), la luz de ocupado se tiene que encender, y la luz de libre se tiene que apagar. Cuando el contador vuelve a bajar, quedando espacio libre para aparcar, entonces la luz de libre se tiene que encender y el de ocupado apagar.

El contador se puede poner a cero con el pulsador de borrar.

4.1.2. Método de implementación de la práctica

El proceso de simulación de la práctica del parking es recrear el movimiento de la barrera, tanto de salida como de entrada del parking. Para ello, he empleado fotogramas que moviéndolos de manera ordenada simulan dicho movimiento. El tiempo entre fotogramas, lo controlo por los ciclos de ejecución global de la aplicación que corre en LabVIEW.

Los fotogramas se mueven cuando la variable de "Targ_1" esta activa y la barrera no ha llegado a final de carrera, en el caso de recrear la entrada del parking.

He logrado recrear los coches mediante el uso de deslizador y he cambiado el cursor del Deslizador por la fotografía de un coche, así puedo controlar en LabVIEW en que lugar de la entrada o salida del parking esta situado el coche.

Los pulsadores que imitan las tarjetas tanto de entrada como de salida, como los indicadores que señalan tanto la situación de las barreras, como el semáforo de paso del parking, y el control numérico del estado del parking, como también los sensores de posición son propios del programa LABVIEW.

4.1.3. Detalles de programación

Cuando estamos moviendo el coche con el deslizador el programa LabVIEW se para hasta que no hemos situado el coche donde queremos, con esto quiero decir que el programa LabVIEW no responde a la comunicación, no corre el ActiveX de CX-Server lite.

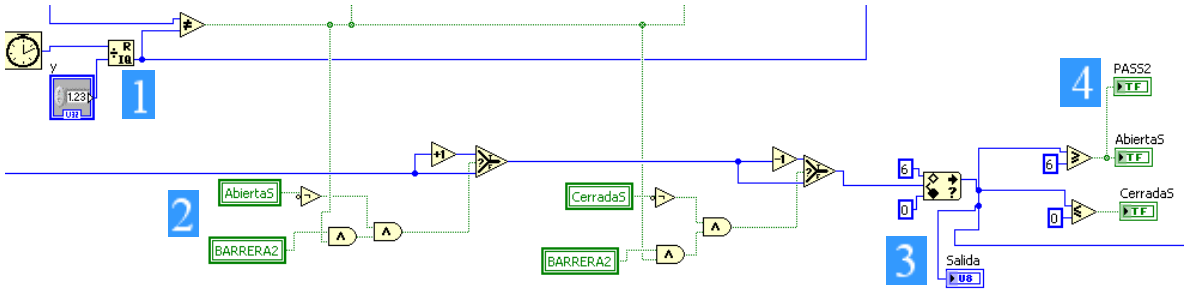


Fig. 42. Algoritmo de las barreras.

Este es el esquema del algoritmo de programación para mover la barrera de salida, es igual al esquema de programación para mover la barrera de entrada.

Es la parte mas complicada de esta aplicación, ya que la secuencia de mover la barrera viene dada por las entradas de los sensores de la barrera y sobretodo por los ciclos que hace el programa LabVIEW cuando se ejecuta y, por lo tanto es quien indica la velocidad de las imágenes para simular el movimiento de la barrera. Fíjense que hemos colocado un control de tiempo para controlar la velocidad. Y, por consiguiente, para saber cuando puedo dar luz verde al coche. Para que pase por la barrera he colocado un indicador el cual nos indica cuando a imagen esta situado cuando la barrera esta en alto.

A continuación indicamos las partes más importantes del algoritmo:

1. Aquí tenemos el control de tiempo de la simulación del movimiento de la barrera.

2. Esta dos variables son los sensores que activan el movimiento de la barrea.

3. Esta variable es donde están las imágenes que simulan el movimiento.

4. Es la variable que da luz verde al coche para que entre al parking si estamos en la entrada del mismo y que para que salgo del parking cuando estamos dentro de él.

4.2. APLICACIÓN 2. BOTELLAS

En la siguiente aplicación recreamos una cadena de llenar y tapar botellas, donde el estudiante podrá entrenar el paralelismo estructurado.

Dicha práctica es una de las mas complicadas a la hora de crear el algoritmo en lenguaje G para simular dicha planta, porque el usuario a penas interactuar con la planta y solo tiene que cargar el algoritmo y ver como funciona. Y, a parte, en este ejercicio, para recrear el máximo posible a una panta real he tenido que colocar unos fallos por si la programación en el PLC está mal implementada.

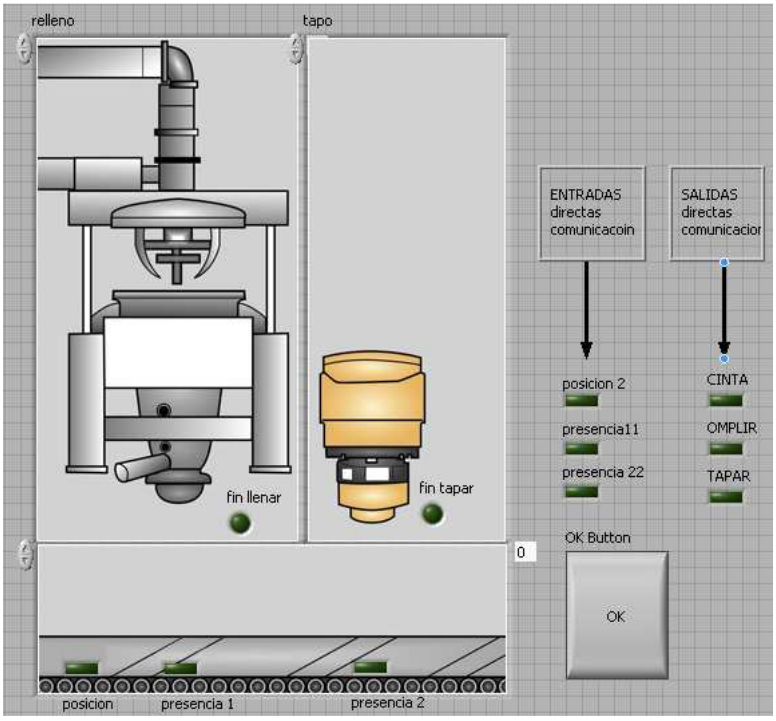


Fig. 43. Panel frontal de la práctica de las botellas.

4.2.1. Funcionamiento

Una cinta transportadora avanza paso a paso y traslada las botellas vacías que serán llenadas y tapas en las diferentes estaciones de trabajo. El sensor de presencia detecta la botella mediante los tacos que preceden las botellas (las botellas pueden aparecer a distancias no constantes). Para cada paso (la longitud recorrida es equivalente a la base de los tacos) se detecta si hay una botella. En caso de presencia de la botella, cada estación ejecutará su tarea, de manera que si no hace nada esperará hasta el próximo paso. También existe un pulsador de puesta en marcha de la cinta transportadora y de los equipos de trabajo.

Por lo tanto, su recreación sería que el pulsar el botón de puesta en marcha el automatismo tiene que funcionar y, entonces la cinta transportadora se tiene que mover paso a paso hasta que detecta una botella.

Si hay una botella para llenar o tapar los sensores de presencia lo detecta y harán sus respectivas tareas hasta que los sensores de fin llenar o fin tapar se activen.

4.2.2. Método de implementación de la práctica

La idea principal era colocar solo un contenedor de imágenes pero nos salían muchas imágenes en el contenedor de imágenes. Por lo tanto, he considerado partir la imagen en tres contenedores, una para cada acción que hace la planta, es decir, una para cada salida del autómata que controla, y así conseguimos una simplificación tanto en el numero de imágenes, como también, una mejora a la hora de programar la planta en lenguaje G.

Cada imagen es controlada por el autómata, es decir, una para la cinta transportadora, otra para la estación de llenado, y otra para la estación de tapar.

La cinta transportadora lleva las botellas, y están separadas una distancia tal que coincidan con el sensor de cada estación. Las imágenes de la cinta esta pensado para que en una de sus situaciones falten una botella en la estación de tapar cuando si hay una botella para llenar. Como también, hay otra situación que no hay una botella en la estación de llenar y si en la estación de tapar.

A continuación, expongo la simulación de los fallos cuando la programación en Cx-Programmer ha sido mal diseñada.

a) Que en la estación de llenado se active sin que haiga un bidón para llenar:

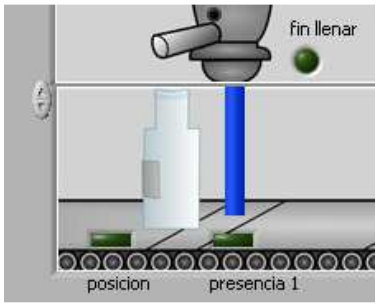


Fig. 44. Simulación de la estación de llenado sin botella.

b) También hemos hecho lo mismo para la estación que tapa los bidones. En este caso, reproducimos como se cae el tapón cuando no hay bidón a cerrar.

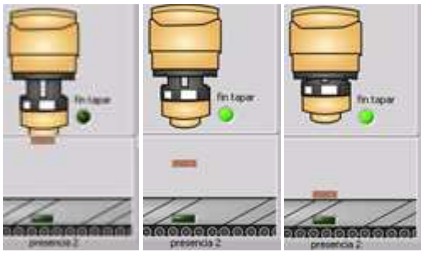
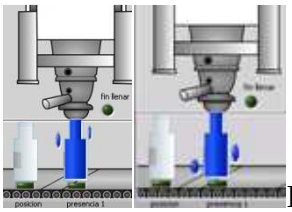


Fig. 45. Secuencia del fallo cuando ni hay botella a tapar.

c) En el caso que un bidón se quede en una estación de llenado mas tiempo de lo debido o que se quede parado, repitiendo la acción de llenar.

 Aquí la manguera de llenado esta arriba por que ya ha hecho su función

 En esta secuencia la manguera de llenado vuelve a bajar otra vez

 En esta secuencia vemos como el líquido se desborda y seguirá así hasta que no se retira la manguera

 Y cuando la cinta se vuelve a mover el bidón se ve totalmente lleno

d) Lo mismo sucede con la estación de tapar cuando un bidón se queda parado y repetimos la acción más de una vez.

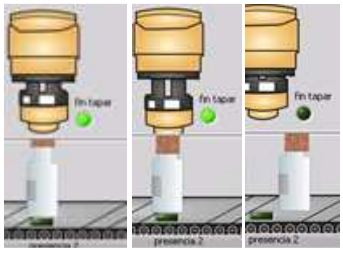


Fig. 46. Secuencia del fallo cuando en la estación de tapar.

Como se puede ver en esta secuencia cuando se repite la acción de tapar en más de una vez se visualiza un doble taponamiento en el bidón.

4.2.3. Detalles de programación

Uno de lo problemas que tuve en este ejercicio, a parte de realizar la programación para recrea los fallos de una planta mal implementada es el movimiento de las botellas cuando están llenas, ya que la reproducción del líquido es un indicador tipo tanque que lo habilito y lo deshabilito al paso de la cinta. También hago lo mismo con el tapón el cual es un simple indicador.

El problema surge cuando tengo que mover tanto la cinta que lleva las botellas como los tanques que hacen de líquido y los tapones a la vez y que se vea todo en conjunto.

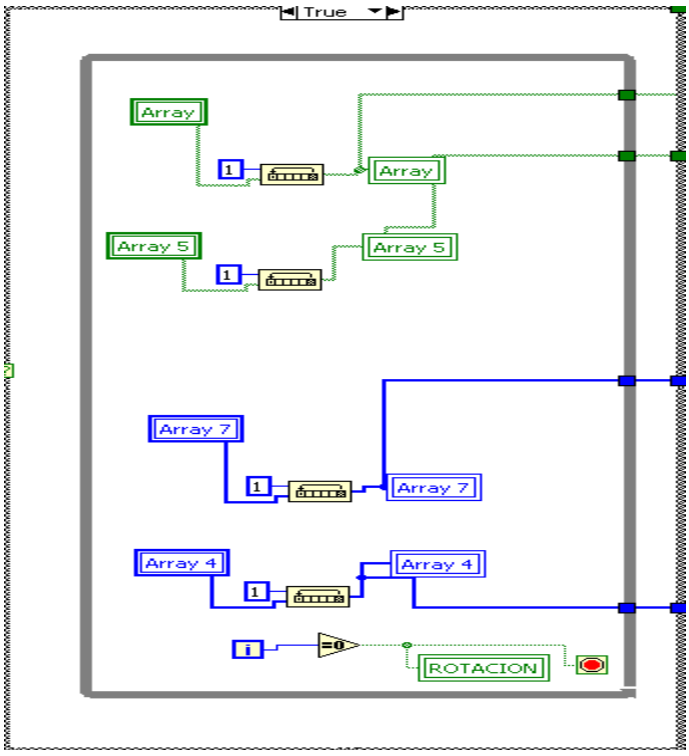


Fig. 47. Algoritmo en paralelo de los estados de las botellas.

Para ello trabajo en paralelo mediante una tabla. Las tablas 5 y 4 son los estados del tapón y del líquido del bidón respectivamente.

LabVIEW no puede trabajar en paralelo. La ejecución de su código es secuencial, no obstante, para hacer lo mas aproximado a una ejecución en paralelo es englobar las variables que quieras que salgan a la vez en una estructura iterativa como un “While”, el cual solo da toda la información cuando salimos de ella, es decir, cuando hay un flag o acción el cual nos hace salir de esta iteración.

4.3. APLICACIÓN 3. TREN

En la próxima práctica se fundamenta en el cruce de raíles de dos trenes, y gestionar el control de los semáforos como el control de la barrera que hay en el cruce. El control de la barrera y los semáforos los gestiona un autómata el cual esta programado para que actúe según los sensores que hay en las vías.

Fig. 48. Panel frontal del la practica del tren.

4.3.1. Funcionamiento del ejercicio

Al principio de la práctica las barreras estarán levantadas y los semáforos en verdes. Si pulsamos el botón de la "izquierda" el tren de la izquierda se moverá hacia la derecha, y lo hará en sentido contrario el tren de la derecha si pulsamos el botón de la "derecha".

En las vías tenemos dos sistemas de detección **a** y **c** que son los que provocan el cierre de las barreras y la activación de los semáforos que se pongan en rojo.

Cuando uno de los trenes deja de pasar por el sensor **b**, entonces las barreras se levantan. Y cuando uno de los trenes pasa por sus respectivos sensores **e** y **f** al final de la vía entonces, los semáforos se ponen en verde para que el otro tren que viene en dirección opuesta pueda entrar en el cruce.

4.3.2. Método de implementación de la práctica

He dispuesto dicha aplicación como la práctica de las botellas, es decir, he separado una imagen en partes, en este caso, cinco partes, una para cada sensor, cada cual es independiente una de la otra. Este método de dividir las imágenes agiliza la programación de la planta en LabVIEW.

Cuando clicamos el botón “izquierda”, por ejemplo, habilitamos la orden de mover las imágenes que hemos montado en el contenedor de imágenes, donde está el tren al inicio para que se muevan consecutivamente una tras otra, haciéndonos creer que el tren se mueve de izquierda a derecha.

El tiempo que pasa de imagen en imagen es un múltiplo de los ciclos de la iteración principal del programa que se está ejecutando en LabVIEW.

La simulación de las barreras lo forman dos deslizadores. Como la simulación está diseñada des de una perspectiva aérea, la recreación que hacen los deslizadores de las barreras es como si la proyección de las mismas vista des de arriba hace como si los deslizadores decrecieran dando a entender que la barrera se está levantando.

El único fallo visual que se puede tener en cuenta, es cuando los dos trenes entran al cruce a la vez. Por lo tanto, habrá un choque. Para ello he programado la aplicación, en la cual cuando lleguen los dos trenes al final del contenedor de imágenes de inicio o que unos de ellos este en el carril de cruce, y el otro este a punto de entrar en el cruce, salte una imagen de descarrilamiento entre dos trenes.

4.3.3. Detalles de programación

Como en este ejercicio tenemos dos objetos a controlar que se mueven simultáneamente he tenido que hacer dos programas, dentro de la iteración global, uno para cada tren que se ejecutan a la vez en LabVIEW.

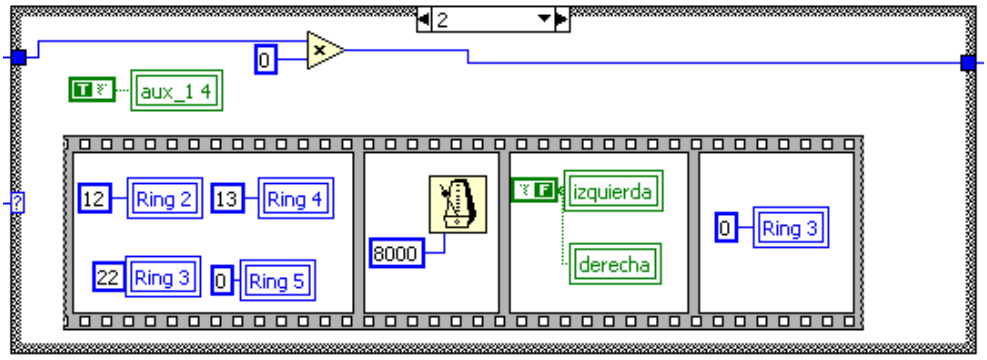


Fig. 49. Algoritmo del tren de la izquierda.

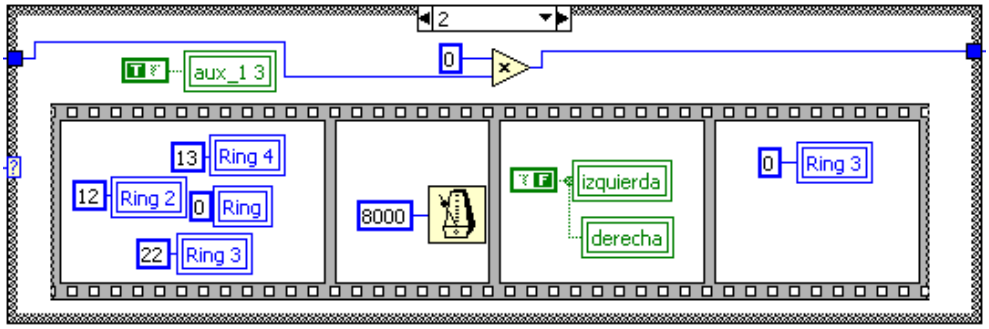


Fig. 50. Algoritmo del tren de la izquierda.

En esta práctica he divido la imagen en cinco partes una para cada sensor. Así como en la practica anterior es mas fácil de montar la programación en G. Y también para reducir el tamaño de las imágenes que tendría que usar en el panel frontal.

4.4. APLICACIÓN 4. PID_MOTOR

En este ejercicio reproducimos el movimiento de un motor DC, el cual lo controlan mediante el PID del autómata.

En esta práctica, tenemos que configurar los parámetros del PID que nos da OMRON en el CX-Programmer para controlar un motor continuo.

Fig 51. Panel frontal del la practica del motor DC.

4.4.1. Funcionamiento

Para activar el ejercicio tenemos que pulsar el botón de “validar” para que la planta adquiera los parámetros de configuración. Luego con el curso, el deslizador, indicamos la consigna a seguir. Lo que controlamos es la velocidad del motor, por lo tanto en la consigna, escogemos la velocidad a la que queremos que vaya el motor.

4.4.2. Método de implementación de la práctica

Esta aplicación aun que parezca la más sencilla a la hora de programar la planta, ya que los elementos los tienes las librerías de LabVIEW, es la más completa a la hora de diseñarla. Por que utilizamos módulos que hemos estudiado en asignaturas de control de la carrera de ingeniería técnica industrial, como la función zeta, y también como pasar de una función continúa a discreta.

En el panel frontal hemos insertado un cursor donde indicamos la consigna del sistema.

4.4.3. Detalles de la programación

En este ejercicio he usado temario que me han enseñado durante el curso tanto en regulación automática como en teoría de control.

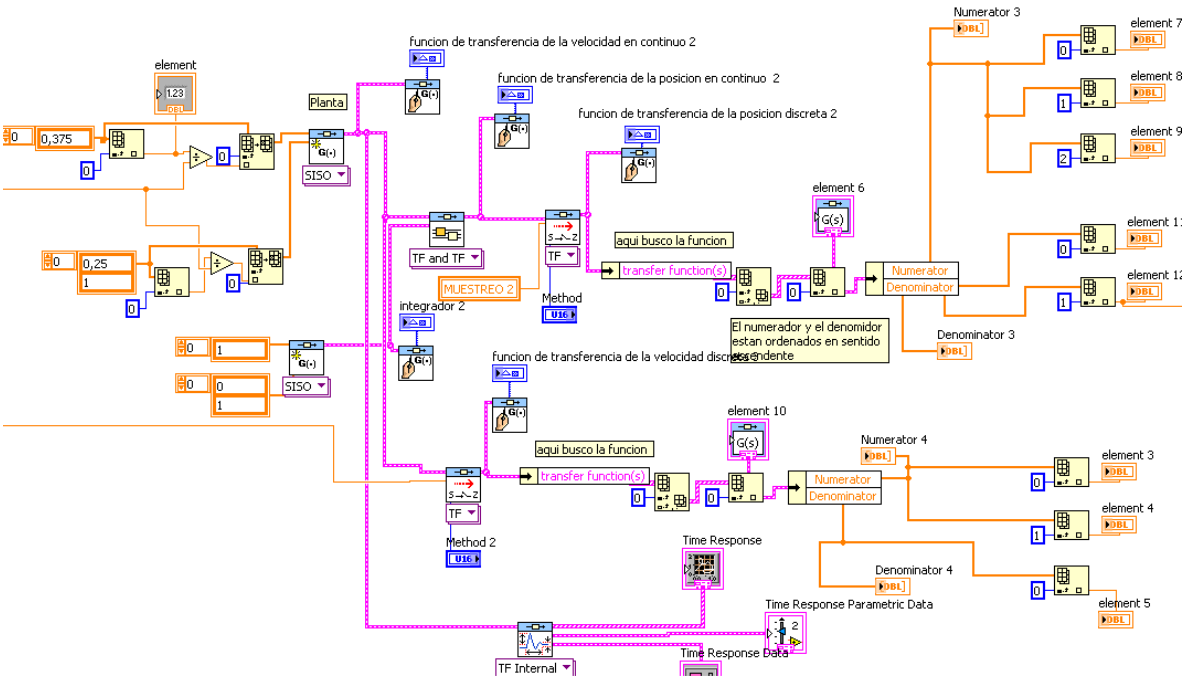


Fig. 52. Conversión de Laplace a función Zeta en el diagrama de bloques..

En este algoritmo, en lenguaje G, se puede ver como uso las librearías de discretización, es decir, como paso las variables continuas del motor a discretas para poderlas procesar en LabVIEW.

En este diagrama lo que conseguimos es pasar de analógico a digital los valores continuos del sistema.

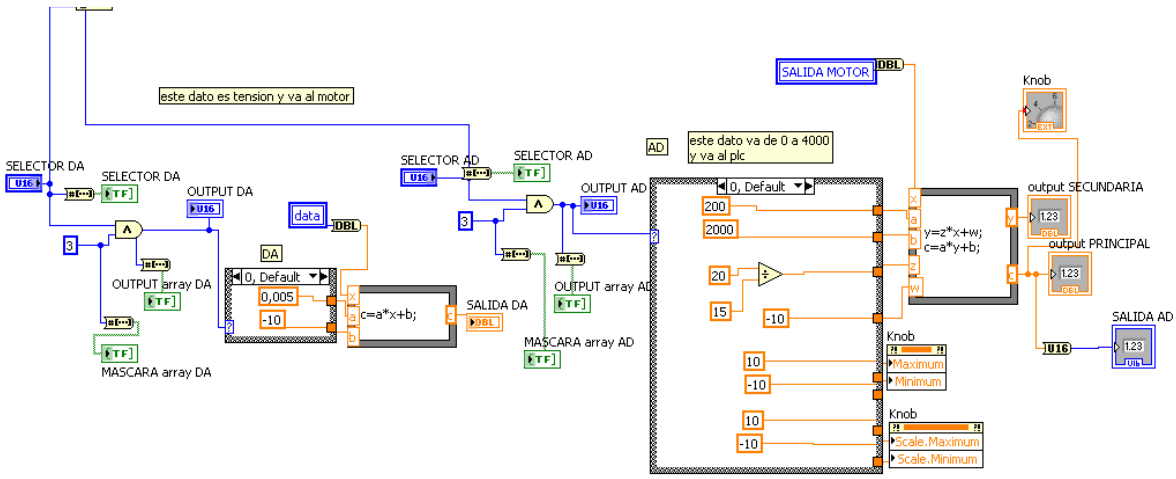


Fig. 53. Conversión de puntos a tensión y viceversa.

Como podemos ver en este diagrama de bloques pasamos la función discreta a valores continuos para que se puedan ver en el motor.

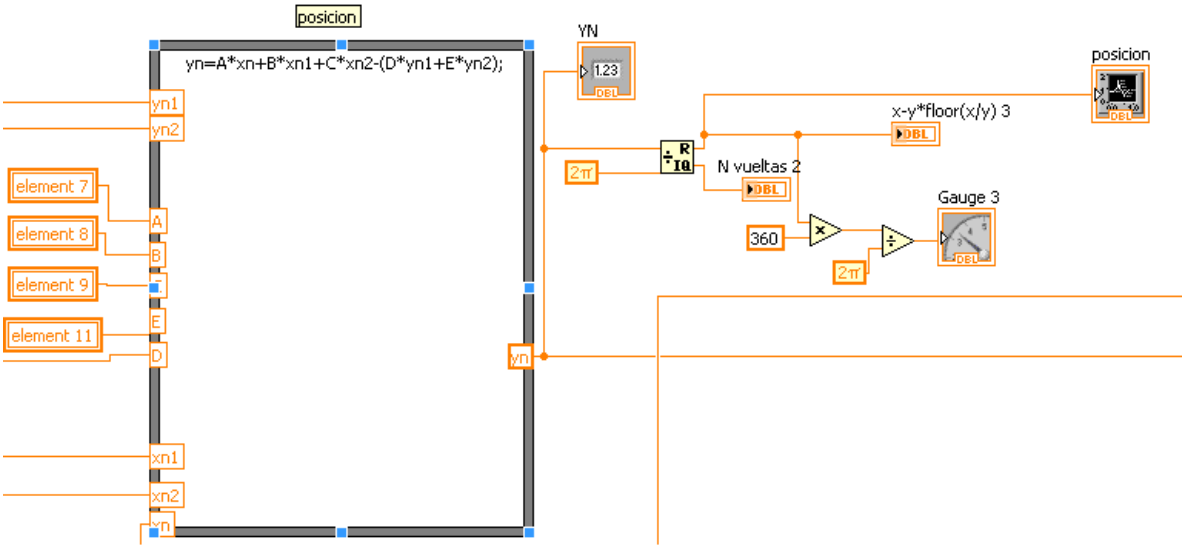


Fig. 54. Serie de potencias de la posición del motor.

En este ultimo diagrama de bloques lo que obtenemos es la velocidad del motor mediante la formula node que nos da LabVIEW.

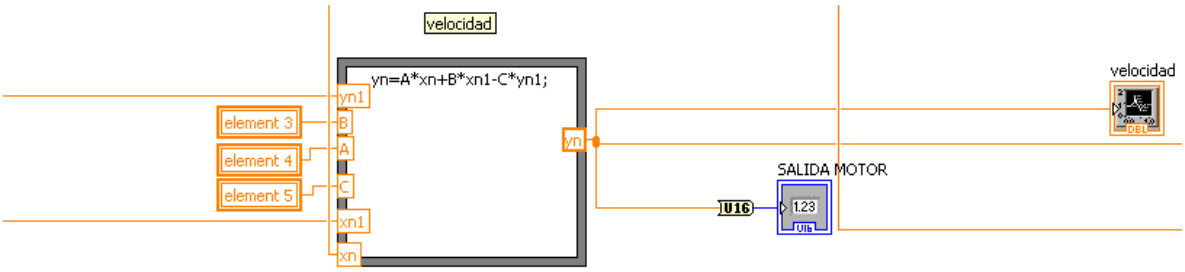


Fig. 55. Serie de potencias de la velocidad del motor.

4.5. APLICACIÓN 5. BOLERA

En este ejercicio recreamos una bolera, mas bien, el funcionamiento de la maquina de bolos.

Solo se pueden dejar de pie un máximo de tres bolos si no se haría muy complejo, aparte que es suficiente para simular la bolera.

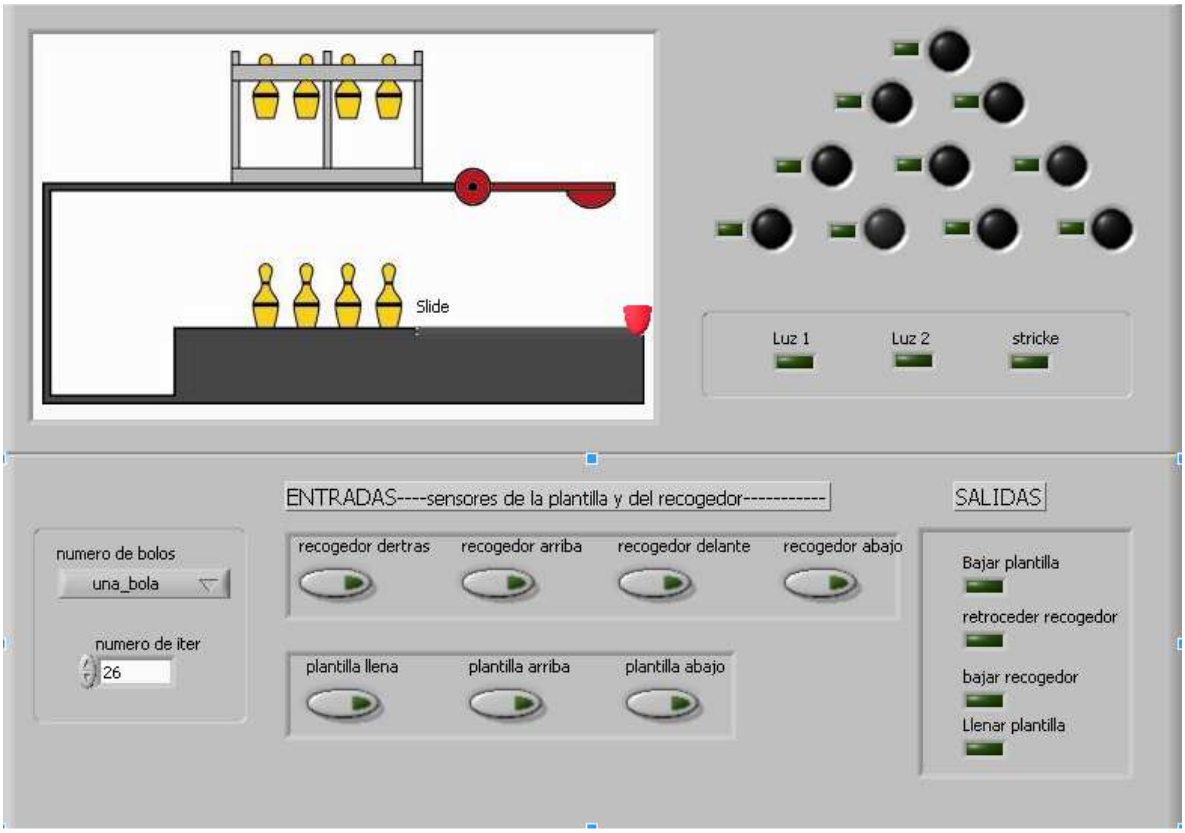


Fig. 56. Panel frontal del la practica de la bolera.

4.5.1. Funcionamiento

En condiciones iníciales, las bolos están colocados en la pista, la plantilla de soporte de los bolos está arriba, el recogedor esta arriba y delante y, la luz de primera tirada esta encendida.

En la primera tirada cuando la bola llega a los bolos se activa el temporizador durante 10 segundos para dejar tiempo “real” para que caigan los bolos y se recojan. Pasado este tiempo se indican los bolos que han caído, y se apaga la luz de la primera tirada, entonces baja la plantilla para coger los bolos que han quedado de pie, cerramos las pinzas para recoger los bolos y subimos la plantilla. Después bajamos el recogedor para retirar los bolos que han caído, y mas tarde se sube el recogedor hasta la posición de reposo. Por último, baja la plantilla para colocar los bolos que no hemos tirado, abre las pinzas para soltar los bolos que han quedado de pie. Por consiguiente,

subimos la plantilla y encendemos la luz de primera tirada y segunda tirada para indicar al usuario que vuelva a tirar.

En la segunda tirada procedemos como en el apartado anterior, cuando hemos tirado la bola, el recogedor recoge los bolos que han caído, retrocede y sube hasta su posición de reposo. Y finalmente, damos la orden de llenar la plantilla y cerramos las pinzas, mas tarde, baja la plantilla para dejar los bolos cuando abre las pinzas, entonces vuelve a subir.

Por ultimo, si hacemos un strike, activamos la luz durante 10 segundos y, hacemos el movimiento de la segunda tirada.

4.5.2. Método de implementación de la práctica

Esta fue unas de las primeras prácticas que hice utilizando imágenes, en este caso nos salieron mas imágenes de lo normal porque las acciones estaban todas juntas en la misma imagen.

El proceso consta de la bola que tira los bolos, la cual, es un DESLIZADOR que hemos modificado el cursor por una imagen de un bolo y un contenedor de imágenes que simulan el movimiento que hacen los bolos al caer cuando tiramos la bola, y las acciones que hacen la máquina de recoger y colocar bolos.

El contenedor de imágenes tiene una secuencia de imágenes ordenada numéricamente, para que si la programación del autómata es la correcta, entonces su funcionamiento es lineal, es decir no da saltos.

Como he comentado antes, la implementación de este ejerció empleando solo una imagen para todo el proceso, me ha llevado a recrear los fallos de forma diferente. Es decir, lo que hago es informa en el contenido de imágenes el error en el proceso que se ha cometido.

Fallos de programación en el autómata para la planta de la bolera:

- Que el usuario has bajado el recogedor cuando tenia que bajar la plantilla.
- Ha bajado el recogedor cuando la plantilla esta abajo.
- Ha bajado la plantilla cuando tiene el recogedor abajo.
- El recogedor se mueve ha activado bajar la plantilla.
- La plantilla esta llena y ha dado la orden de llenarla.
- Esta bajando el recogedor y llenando la plantilla a la vez.
- Estas bajando el recogedor y la plantilla a la vez.
- El recogedor esta retrocediendo cuando tenias que bajar la plantilla.

- Baja la plantilla y retrocede el recogedor cuando debías de bajar solo la plantilla.
- Llenas la plantilla cuando solo tenía que bajarla.
- Baja la plantilla y la llenas a la vez cuando solo la tiene que llenar.
- Baja el recogedor y llena la plantilla cuando tenías que bajar la plantilla.
- Baja la plantilla y la llenas a la vez, y también retrocede el recogedor cuando solo tenías que bajar la plantilla.
- Retrocede recogedor y llenas la plantilla a la vez cuando solo tiene que bajar la plantilla.
- Retrocede recogedor mas llena la plantilla, y también da la orden de bajar cuando solo tienes que bajar la plantilla.
- Retrocede el recogedor mas llena la plantilla, también baja el recogedor cuando tiene que bajar la plantilla.
- Ha activado todas las acciones cuando solo tiene que bajar la plantilla.
- Baja el recogedor cuando tiene que subir la plantilla.
- Baja la plantilla y el recogedor cuando la plantilla está abajo y la tiene que subir.
- Retrocede y baja el recogedor cuando tiene que subir la plantilla.
- Baja plantilla y retrocede el recogedor cuando tiene que subir la plantilla.
- Baja y retrocede el recogedor cuando tiene que subir la plantilla.
- Baja la plantilla, retrocede el recogedor y también esta activado la acción de retroceder cuando tiene que subir la plantilla.
- Llenar la plantilla cuando tiene que subir.
- Baja y llena la plantilla cuando tiene que subir.
- Baja recogedor y llena la plantilla cuando la plantilla tiene que subir.
- Baja la plantilla y llena, y a demás tiene activa la acción de retroceder el recogedor cuando tiene que subir la plantilla.
- Retrocede el recogedor y llena la plantilla cuando tiene que subir la plantilla.

- Retrocede y baja el recogedor y llena la plantilla cuando tiene que subir la plantilla.
- Todas las acciones activas cuando tiene que subir la plantilla.
- Baja la plantilla cuando tiene que bajar el recogedor.
- Baja la plantilla y el recogedor cuando tiene que bajar el recogedor.
- Retrocede el recogedor cuando tiene que bajar.
- Baja plantilla y retrocede el recogedor cuando tiene que bajar el recogedor.
- Baja el recogedor y retrocede a la vez cuando solo tiene que bajar.
- Baja la plantilla, baja el recogedor y retrocede cuando solo tiene que bajar el recogedor.
- Llenar l aplantilla cuando tiene que bajar el recogedor.
- Baja la plantilla y tiene activa la acción de llenar a la vez cuando tiene que bajar el recogedor.
- Retrocede el recogedor, la plantilla se llena y baja a la vez cuando tiene que bajar el recogedor.
- Todo activo y solo tiene que bajar el recogedor.

Fig. 57. Ventana que sale en el contenedor de imágenes indicando el tipo de error

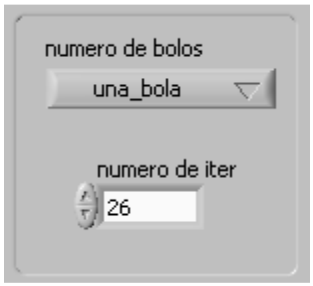


Fig. 58. Control del número de iteraciones y selección de bolos caídos.

4.5.3. Detalles de programación

Lo más difícil de esta práctica es como se mueven las imágenes según las entradas del sistema. Para ello he tenido que hacer un “case” para cada caso por que no he partido la imagen, y por lo tanto, no he podido asociar cada imagen a cada variable de salida a controlar.

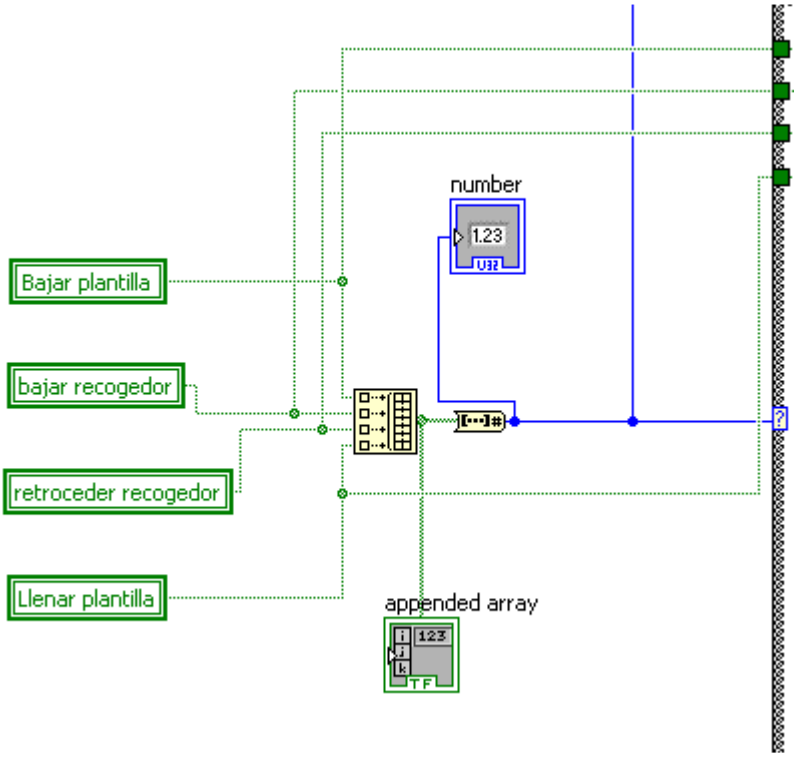


Fig. 59. Algoritmo que actúa según la acción seleccionada

Como solo se puede ejecutar una acción, cuando dos acciones o mas se activan a la vez pasamos al mecanismo de errores.

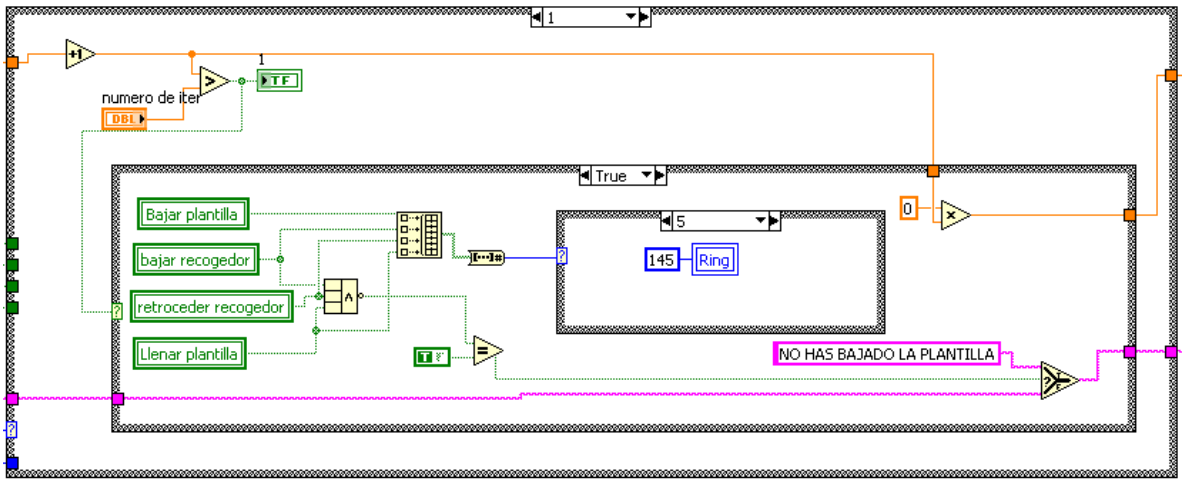


Fig. 60. Algoritmo de fallos.

4.6. APLICACIÓN 6. MEZCLADORA

Nuestra aplicación se basa en una instalación mezcladora que tiene dos depósitos los cuales contienen dos productos A y B que se vacían alternadamente sobre un recipiente C que hace de báscula, así podemos seleccionar la cantidad de cada uno de los productos que pasará a mezclarse.

El mezclador M permite obtener la mezcla formada por estos dos productos gracias a la rotación de una hélice.

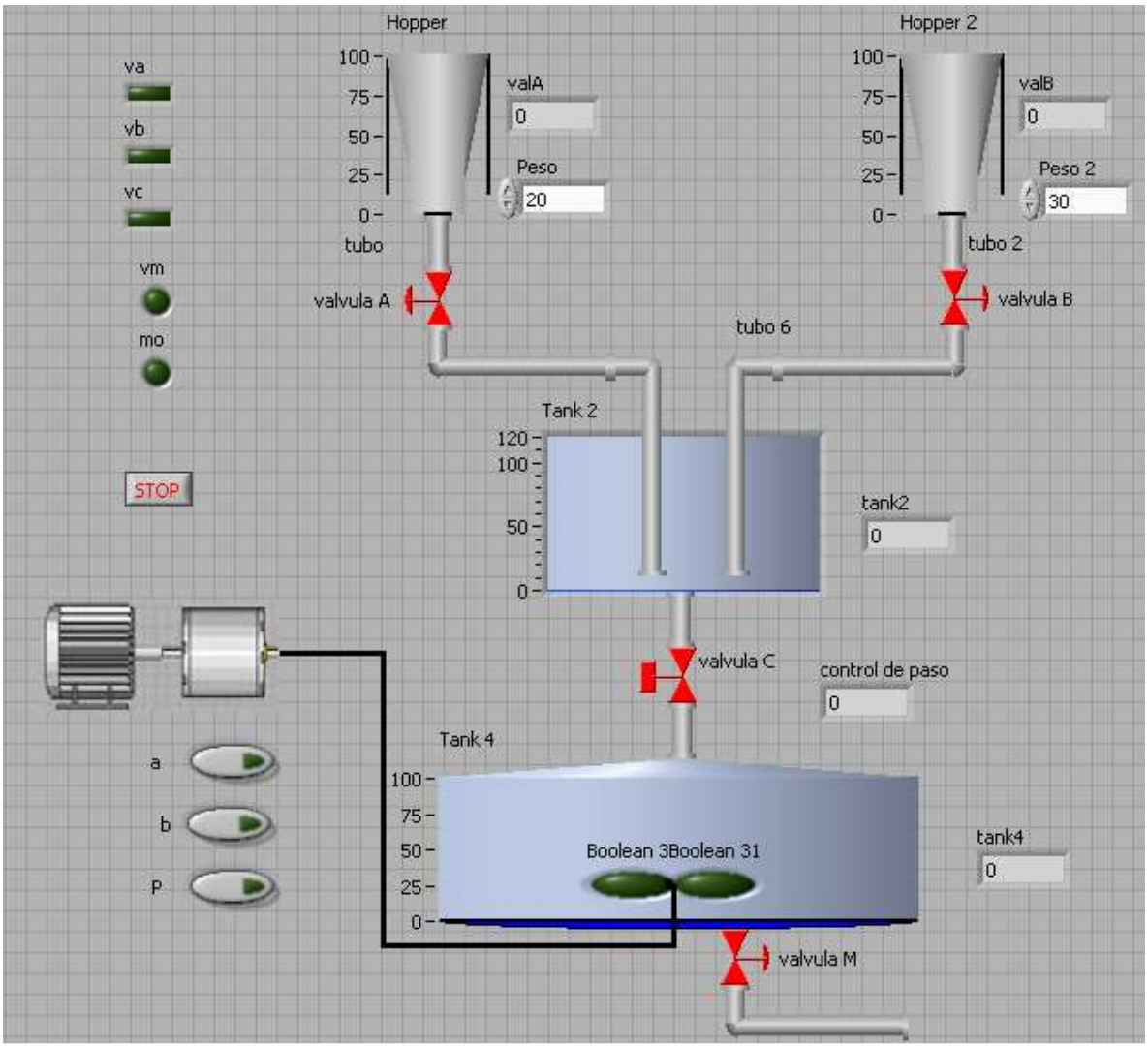


Fig. 61. Panel frontal del la practica de la mezcladora.

4.6.1. Método de implementación de la práctica

En este ejercicio he dispuesto de cuatro tanques. Los tanques superiores se llenan con un controlador numérico sencillo y decrecen cuando se activan sus respectivas válvulas. Como el nivel de los tanques tienen un numero asociado, sólo es una cuestión aritmética para llenar el siguiente tanque lo mismo sucede para llenar el

ultima tanque. Aunque, en este caso, solo se tiene que pasar el valor del tanque del medio.

El vaciado de los tanques los controlo mediante los ciclos de la iteración global del programa que se ejecuta en LabVIEW.

4.6.2. Funcionamiento de la aplicación

La orden de inicio la dará un operario apretando un pulsador *P* siempre y cuando las condiciones iníciales son ciertas (*C* y *M* vacíos).

Entonces, pesamos la cantidad de producto *A* (abriendo la válvula **va**) en *C* (hasta llegar al peso deseado, dato obtenido mediante el sensor **a**) e inmediatamente volcada al mezclador a través de la válvula **vc** hasta que el recipiente *C* quede vacío (dato obtenido con el sensor **c**).

De igual manera, pesamos la cantidad de producto *B* (abriendo la válvula **vb**) en *C* (hasta llegar al peso deseado, dato obtenido mediante el sensor **c**) e inmediatamente volcada al mezclador a través de la válvula **vc** hasta que el recipiente *C* quede vacío (dato obtenido con el sensor **c**).

Seguidamente se activa el motor de la hélice (**mo**), el producto A y el producto B son mezclados hasta llegar al nivel de mezcla deseado, indicado por el sensor **m**.
Finalmente vaciaremos el contenido del mezclador *M* a través de la válvula **vm** hasta que éste quede vacío.

4.6.3. Detalles de programación

Esta es unas de las prácticas más fáciles de programar en LabVIEW porque no hace falta insertar imágenes. Todos los elementos que disponemos en el panel frontal son elementos de LAVIEW.

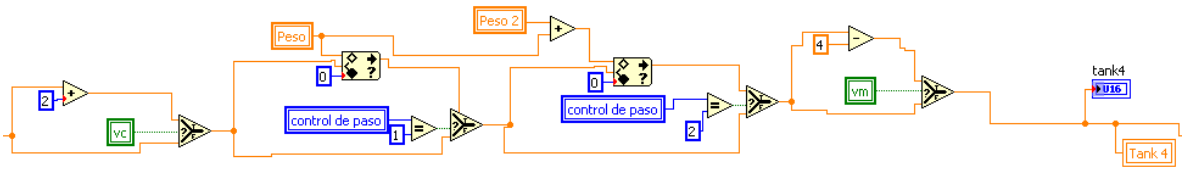


Fig. 62. Algoritmo de simulación del tanque.

La complicación radica en la estructura donde pasa el líquido de los tanques al tanque central, el cual lo tengo que sincronizar para que recree realmente el vaciado al tanque principal.

4.7. APLICACIÓN 7. ASCENSOR

En esta última aplicación vamos a simular un ascensor de 3 pisos, es decir, el control de la planta que recrea un ascensor será gestionado por nuestro PLC:

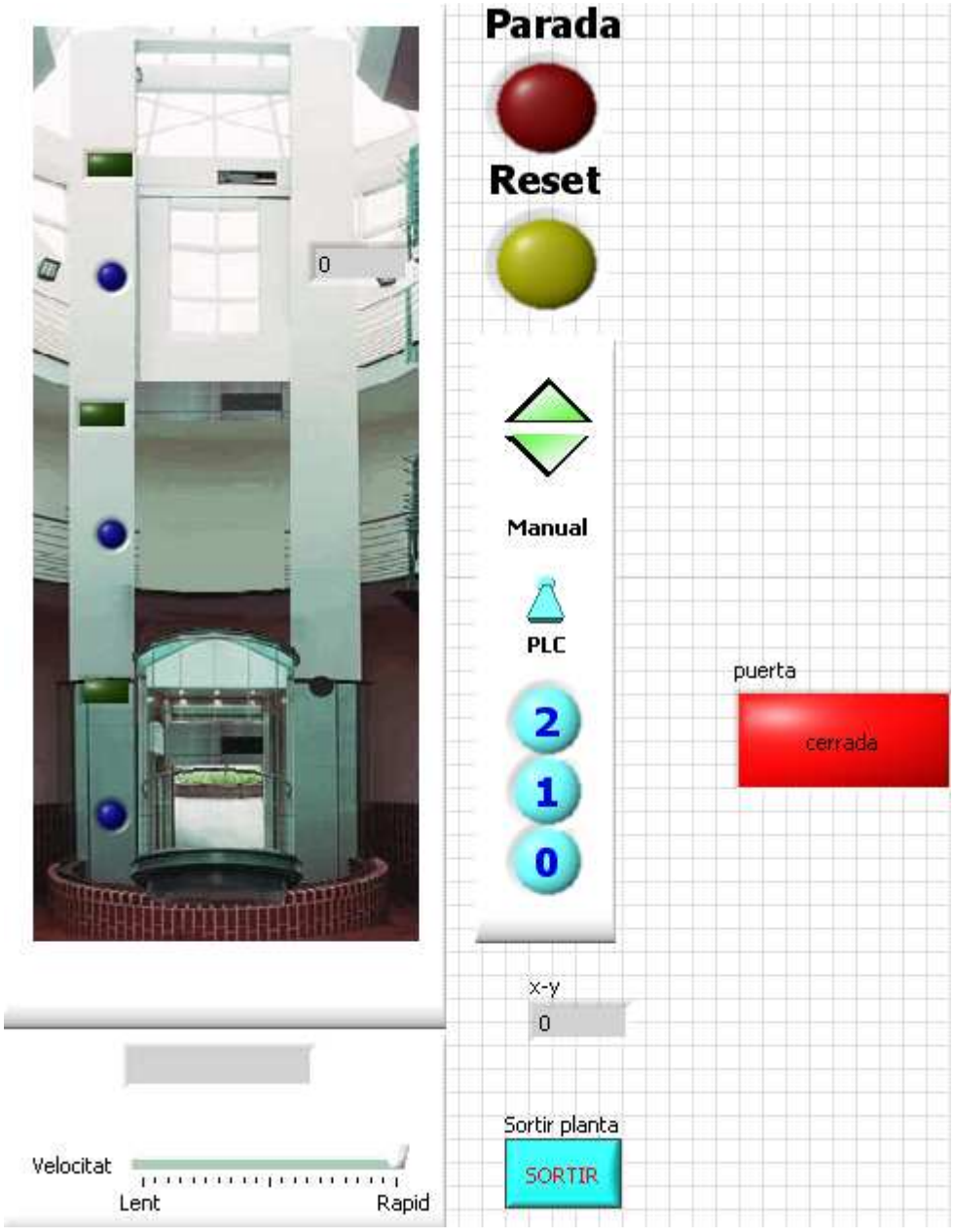


Fig. 63. Panel frontal del la practica del ascensor

4.7.1. Método de implementación de la práctica

La recreación de un ascensor lo he logrado insertando una imagen en la base del deslizador de las plantas donde se mueve el ascensor, y he cambiado el cursor por un ascensor.

Por lo tanto, lo que hacemos es controlar numéricamente la posición del deslizador (elevador) asignado un intervalo de números para cada piso.

4.7.2. Funcionamiento

Para empezar a ejecutar nuestro ejercicio primero tenemos que pulsar el botón de RESET y a partir de aquí entramos en el bucle del programa.

Como se puede observar, nuestro ascensor puede funcionar de forma manual, controlando la posición del ascensor con los marcadores verdes  .O de forma automática, es decir nosotros le indicamos el piso y el PLC se encarga de llevarlo hasta allí.

Por último, se puede controlar la velocidad del movimiento del ascensor con el deslizador horizontal.

4.7.3. Detalles de programación

La parte mas complicada fue diseñar el control del ascensor tanto cuando era manual como cuando también era gestionada por el PLC. En la siguiente ilustración se observa como si al ejecutar la aplicación en la segunda estructura “case” se este subiendo, por eso el símbolo aritmético del “+”

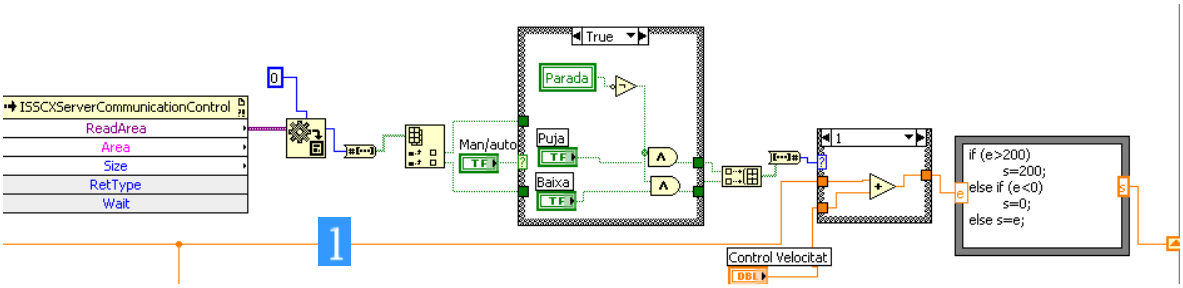


Fig. 64. Algoritmo que controla el movimiento del ascensor manualmente.

Como se puede observa en el diagrama de bloques de LabVIEW. Esta es la parte de la programación donde se implementa el control manual del ascensor. Observe que hemos desconectado las salidas del PLC, que se comunica con la planta mediante el booleano “Manu/Auto” que actúa en dicha estructura “case”. En este caso el selector está en estado manual.

Su funcionamiento seria que cuando el ascensor este parado haga caso a los indicadores de SUBIR y de BAJAR

El siguiente caso seria cuando el control del ascensor es controlado por el PLC:

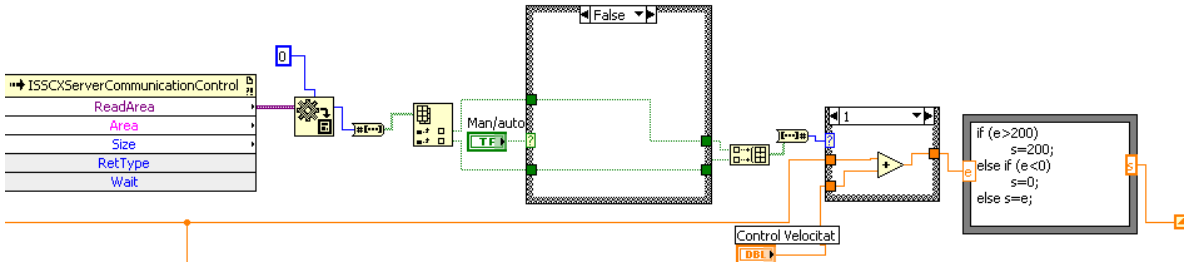


Fig. 65. Algoritmo que controla el movimiento del ascensor automaticamente.

Como podemos ver la salida del PLC está directamente conectada con la planta que simula el ascensor, en este caso el selector esta en “manual”

El ultimo bloque es una formula NODE que nos da el programa labivew donde podemos hacer un pseudo programa muy simple para que se cumplan unas reglas sencillas.

CAPITULO 5. CONFIGURACION DEL PLC

En el próximo capitulo está dedicado íntegramente a la configuración del PLC virtual para poder comunicarse con la planta como también a la configuración del programador. Asimismo hago unas pequeñas reseñas sobre los objetivos de la programación en CX-Programmer de cada práctica.

5.1. PROGRAMACIÓN MEDIANTE CX-PROGRAMMER

CX-Programmer es el programador de los autómatas programables de Omron. Permite programar todos los modelos, desde micro-PLC hasta la nueva serie CS de gama alta. Además de un entorno de programación exhaustivo, CX-Programmer proporciona todas las herramientas necesarias para proyectar, probar y depurar cualquier sistema de automatización.

5.1.1. Crear proyecto con CX-PROGRAMMER

CX-Programmer es un programa que se encarga de implementar algoritmos asociados a los modelos de PLCs OMRON, por lo tanto, es necesario configurar el tipo de PLC donde vamos a volcar el programa y la clase de comunicación con el terminal.

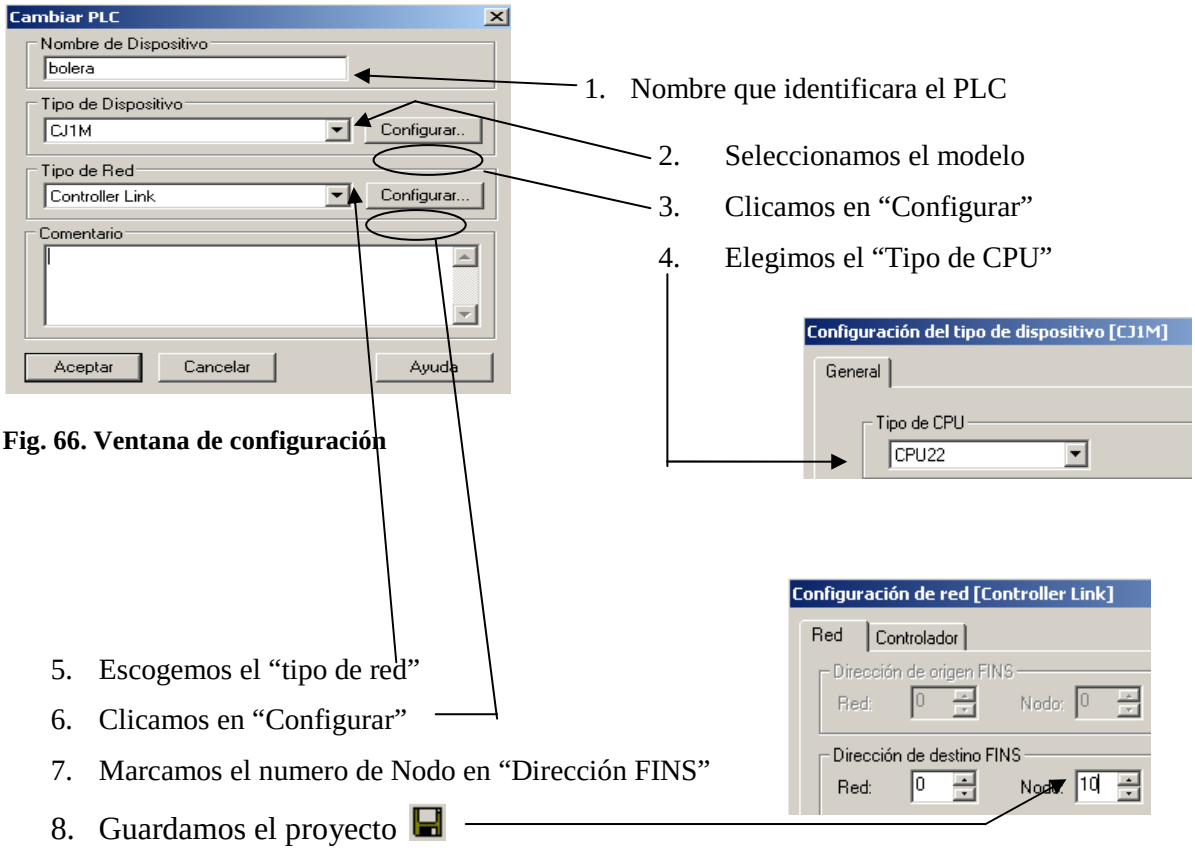


Fig. 66. Ventana de configuración

Nota: El nombre y el contenido del proyecto es diferente del de la CPU. En un Proyecto se pueden insertar diversas CPUs

5.1.2. Crear símbolos en CX-Programmer

Lo primero, antes de comenzar a introducir el código hay que definir los símbolos que serán usados en el programa. Un símbolo no es más que una dirección de memoria a la que se asocia un nombre o un comentario.

Los símbolos pueden ser locales o globales. Los símbolos locales sólo pueden ser usados en el programa en que son definidos. Por otro lado, los símbolos globales definidos para un controlador pueden ser utilizados por cualquiera de sus programas.

Los símbolos locales y globales son almacenados en las tablas de símbolos locales y globales respectivamente. En estas tablas se pueden insertar, borrar, reeditar, los símbolos.

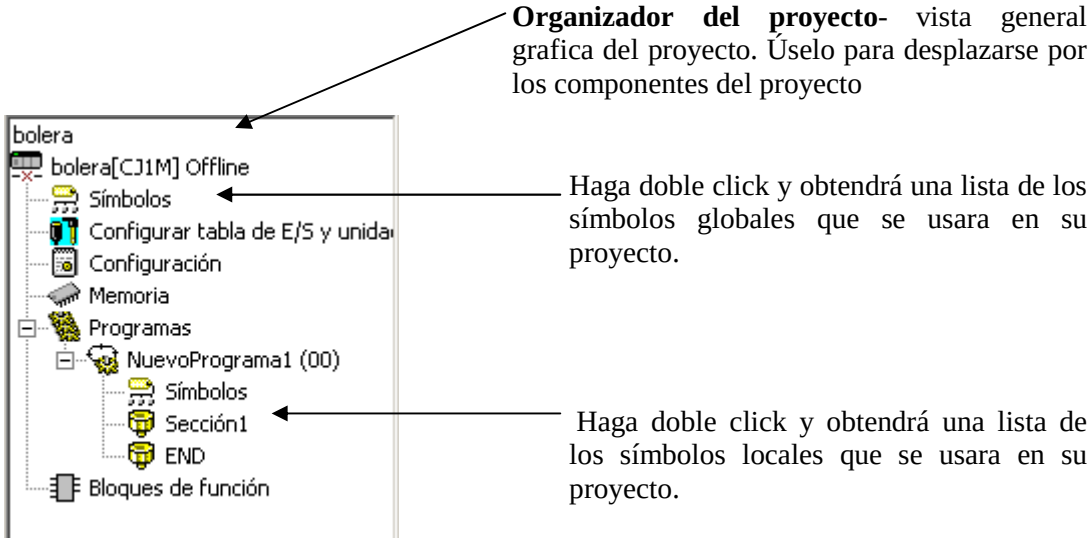


Fig. 67.Ventana de organizador de proyecto

Nota: En la tabla de símbolos globales están definidos por defectos varios de símbolos de uso específico.
La tabla de símbolos locales está totalmente vacía. Es en ella donde definiremos los símbolos a usar en nuestro programa.

Definir los símbolos:

Para definir un símbolo solo tenemos que llenar tres campos, es decir, indicar su nombre, ubicarlo en una dirección e indicarle un tipo de variable. En la tabla siguiente le indicamos los tipos de datos:

- **BOOL**: Variable de un bit
- **CHANNEL**: Variable de una palabra. Hace referencia a cualquier variable no booleana

- **DINT**: Variable de dos palabras en binario con signo
- **INT**: Variable de una palabra en binario con signo.
- **LINT**: Variable de cuatro palabras en binario con signo.
- **NUMBER**: Constante numérica en formato decimal.
- **REAL**: Variable de 2 palabras (32Bit) con formato en coma flotante
- **LREAL**: Variable de 4 palabras (64Bit) con formato en coma flotante
- **UDINT**: Variable de dos palabra en binario sin signo
- **UDINT_BCD**: Variable de dos palabras en formato BCD (8 dígitos).
- **UINT**: Variable de una palabra en binario sin signo
- **UINT_BCD**: Variable de una palabra en formato BCD (4 dígitos).
- **ULINT**: Variable de cuatro palabras en binario sin signo.
- **ULINT_BCD**: Variable de cuatro palabras en formato BCD (16 dígitos)
- **WORD**: Variable de un cadena binaria de 16 bits
- **DWORD**: Variable de un cadena binaria de 32 bits
- **LWORD**: Variable de un cadena binaria de 64 bits
- **STRING**: Los datos que operan como conjuntos de caracteres

1. Haga doble click sobre el icono de símbolos locales 

para visualizar la tabla.
2. En cualquier zona de la tabla hacemos click con el botón derecho y seleccionamos 

, con lo que se abre el siguiente cuadro:

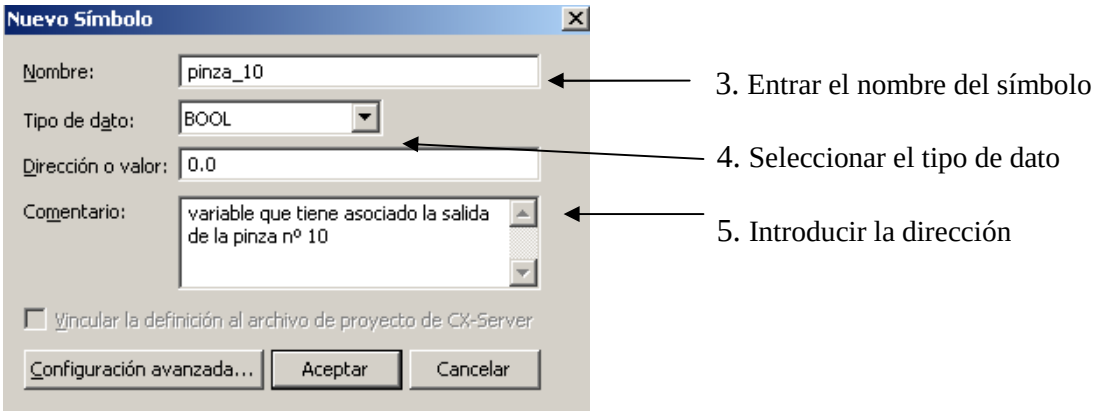


Fig. 68.Ventana donde se defino el punto

5.2. CONFIGURACION DEL SIMULADOR

Con Cx-Simulator se puede conseguir un entorno de depuración equivalente al entorno del sistema PLC real mediante la simulación de la operación en un PLC de la serie CS/CJ en un PLC virtual en el ordenador. Permite la depuración del programa de un PLC antes de montar el sistema real. Reduce el tiempo total necesario para el desarrollo y puesta en marcha de máquinas o equipos.

Fig. 69.Ventana de inicio del simulador

A continuación expondremos los pasos para poder configurar eñ CX-Simulator:

Se nos abrirán dos ventanas, una donde nos da la posibilidad de guardar los parámetros necesarios para la simulación o bien, para cargar los parámetros ya existentes y otra en la cual se ve la interfaz del simulador.

En la consola Debug podemos controlar las acciones, es decir, las líneas de programación que se ejecutan, del autómata de simulación con los comandos que nos viene.

También, en la consola Debug podemos cargar ficheros de configuraciones ya realizadas, sin tener que hacer de nuevo el proceso de configurar, el cual lo expondremos mas adelante.

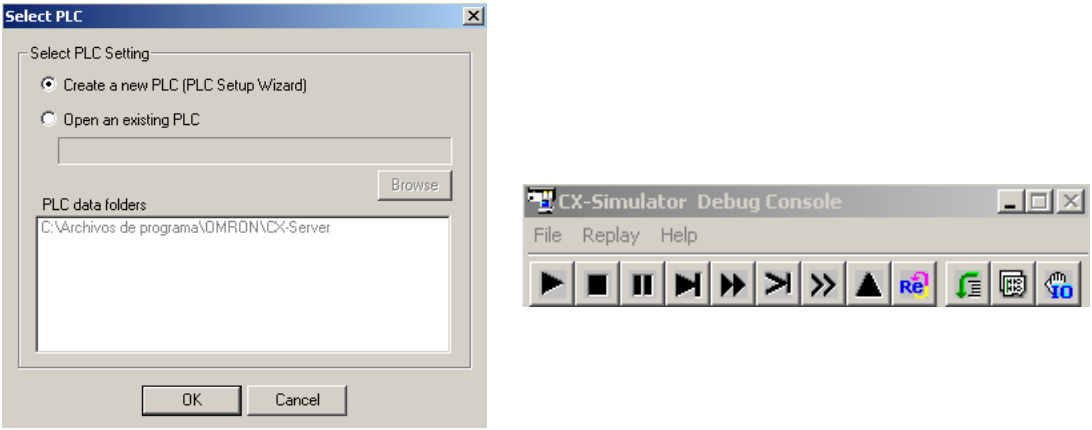
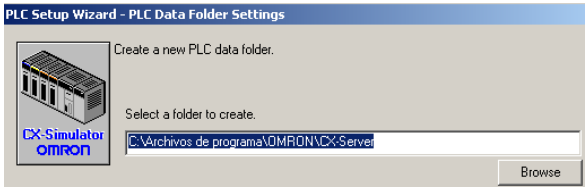


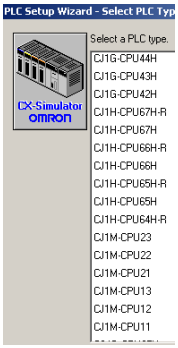
Fig. 70.Ventanas para configurar el simulador.

Si no tenemos la configuración del simulador o queremos cambiar los parámetros ya existentes, entonces marcamos “Create a new PLC”.

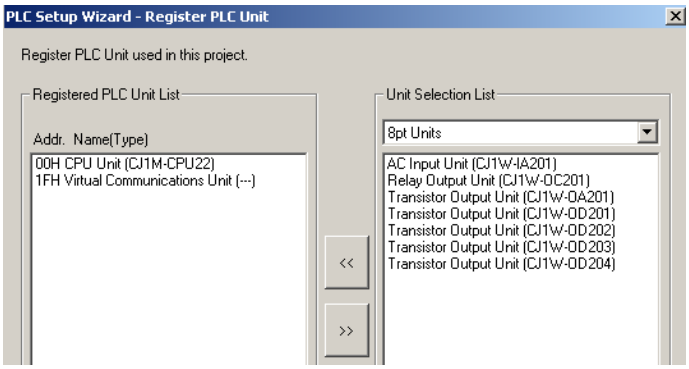
A continuación exponemos la secuencia para configurar el simulador:



1. Elegimos la ubicación donde lo vamos a guardar.

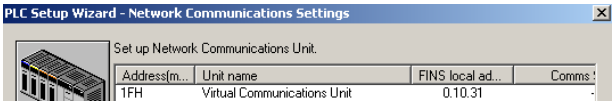


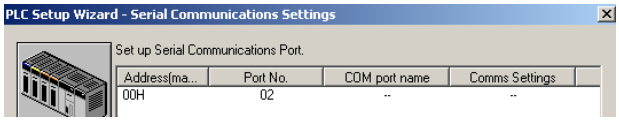
2. En la siguiente ventana seleccionamos el modela de PLC y el modelo de CPU virtual que lo gobierna.



3. Seleccionaremos las unidades I/O que se desea utilizar. Y moveremos de la venta de la derecha a la ventana de la izquierda. Entre las unidades seleccionadas ha de haber por lo menos una unidad de comunicación virtual.

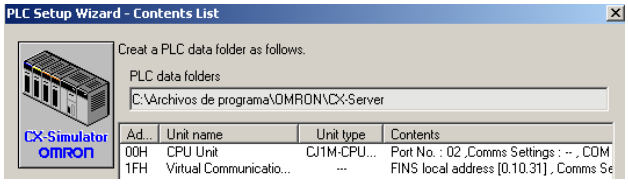
4. En esta ventana configuraremos la dirección FINS de la unidad virtual Edit o lo dejaremos por defecto.





5. En la siguiente ventana configuramos el puerto serie el cual lo dejaremos por defecto.

6. En la siguiente ventana nos informa de la configuración realizada y en la ubicación donde se guarda. Solo nos queda darle Finalizar.



Cuando ya tenemos configurado el simulador y lo encendemos. Se nos aparecerá una ventana donde se aprecia la dirección virtual de comunicación del PLC.

Dicha ventana también se nos aparece cuando ejecutamos el simulador de PLC de OMRON. En ella, se especifica el tipo de comunicación que tenemos en el sistema, como la dirección de red.

Tenemos que pulsar en “connect” para activar el simulador.

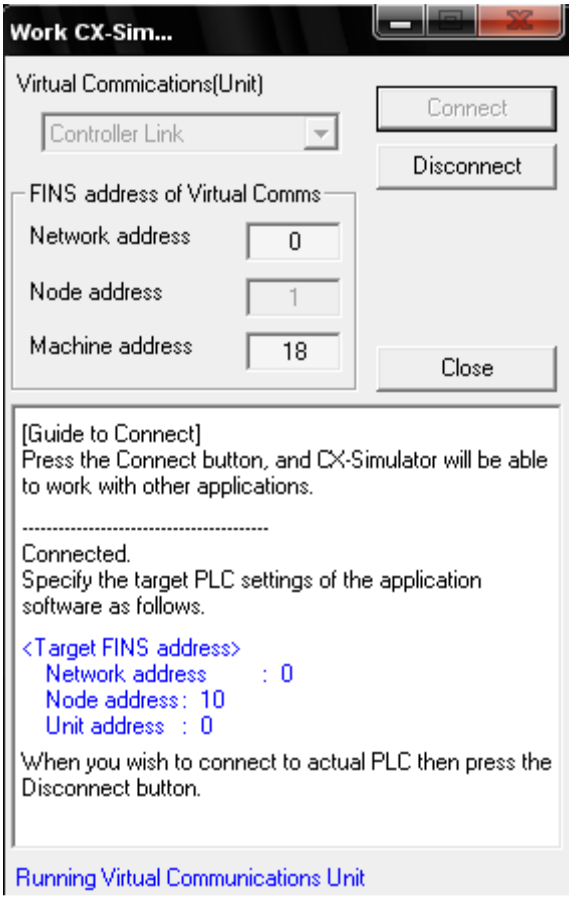


Fig. 71.Ventana de ejecución del simulador

Fijémonos que la dirección de nodo concuerda con la dirección de nodo configurada el el CX-Programmer.

5.3. PROGRAMACION DE LAS PRÁCTICAS CON EL CX-PROGRAMMER

Para diseñar las prácticas de automatización en LabVIEW para los autómatas de OMRON, he tenido que programar con el CX-Programmer. Y verificar su funcionamiento en relación con los objetivos de cada práctica.

Ahora expondré unas pequeñas reseñas de cual es el objetivo de programación de cada práctica:

Aplicación 1 parking: En esta práctica buscamos como el alumno ha adquirido los conocimientos a la hora de programar sistemas con grafcets independientes que comparten variables.

Aplicación 2 botellas: En este ejercicio el alumno practicara el paralelismo estructurado en la programación.

Aplicación 3 tren: En esta actividad, queremos que el alumno conozco como programar una estructura secuencial.

Aplicación 4 PID motor: En este ejercicio buscamos que el alumno sepa configurar un PID de un autómata de OMRON.

Aplicación 5 Bolera: En este practica buscamos que el alumno aprenda ha usar un estructura secuencial compleja.

Aplicación 6 Mezcladora: En esta actividad buscamos que aprendan como se programa un sistema secuencial en cadena.

Aplicación 7 Ascensor: En esta aplicación iniciamos al alumno los grafcets que proporciona el CX-Programmer.

CAPITULO 7.CONCLUSIONES

En este capitulo expongo las dificultades del proyecto, como también las conclusiones que he llegado a la hora de diseñar las practicas.

Tal y como se explica en los objetivos, se pretendía realizar unas aplicaciones prácticas que simularan una planta de automatización lo mas real posible con los autómatas de OMRON, mediante el programa de comunicación CX-Server.

Lograr los objetivos no significa que no se haya encontrado dificultades, sino que han sido varias:

La primera era conocer exactamente como funciona el programa CX-Server Lite bajo la tecnología ActiveX, es decir, como se comunica el autómata con LabVIEW. Tengo conocimientos en la comunicación con servidores OPC, pero en este caso la comunicación es interna del mismo PC y la gestión de los datos queda enmascarada.

La segunda dificultad fue recrear la planta de un proceso en LabVIEW, es decir, en el panel frontal del programa. Los elementos que tenemos no están destinados a la simulación si no a la adquisición y control de datos, por lo tanto, he tenido que adaptar dichos elementos para realizar el proyecto.

La tercera gran dificultad fue optimizar el uso de las librerías de comunicación del CX-Server Lite. Por que a la hora de poner en práctica los ejercicios había un retardo en la comunicación. Tal problema es debido, a que la comunicación entre el PLC y el servidor observe muchos recursos a la CPU del PC. Para realizar las prácticas se necesitan varios programas que funcionen a la vez en el ordenador; LabVIEW que simula la planta, CX-Simulator que actúa como un PLC en nuestro ordenador, y el CX-Programmer con el que implementamos la programación, para que nuestro autómata controle la planta del ejercicio correspondiente. Este último, a la hora de ver la simulación, cuando ya hemos volcado la información en el PLC, es superfluo, pero nos da una idea, en modo monitor, de ver los pasos programados, si son correctos.

La cuarta dificulta, a la hora de diseñar las actividades, fue reproducir los fallos que el alumno puede tener a la hora de programar el autómata en LabVIEW. Tengo que decir que hay prácticas donde no se han introducido fallos, ya que si la programación no es la correcta, la planta haría las acciones erróneas programadas y el alumno se daría cuenta del error.

BIBLIOGRAFIA

Paginas WEB

[1] URL http://www.ni.com [consultado: Ene. / 2011]

[2] URL http://www.mrplc.com [consultado: Nov. / 2010]

[3] URL http://industrial.omron.es [consultado: Dic. / 2010]

[4] URL http://www.infoplc.net [consultado: Nov. / 2010]

[5]URLhttp://industrial.omron.es/es/products/catalogue/automation_systems/software /ru ntime/cx-server_lite/default.html [consultado: Dic. / 2010]

Manuales

[6] SYSMAC CS/CJ/CP Series SYSMAC One NSJ Series - Communications Commands (1999) – OMRON

[7] Autómatas Programables SYSMAC Serie CJ (2001) – OMRON

[8] Autómatas programables Serie SYSMAC CS Serie SYSMAC CJ - Manual de referencia de instrucciones (1999) – OMRON

[9] CX-Server Runtime User Manual Revision 2.4(CX-ONE)

[10] Operation Manual SFC Programming – OMRON (CX-ONE)

[11] CX-Server Runtime User Manual Revision 2.1

Anexo 1

Métodos que contiene el Activex de OMRON en LabVIEW:

- A (get): Función para leer los registros de memoria ubicados en el espacio A.
- A (put): Función para escribir los registros de memoria ubicados en el espacio A.
- About Box: Abre un cuadro de dialogo sobre la información pertinente.
- Active: Devuelve el estado de la conexión de un PLC.
- AR(get): Función para leer los registros de memoria ubicados en el espacio AR.
- AR(put): Función para escribir los registros de memoria ubicados en el espacio A.
- C (get): Función para leer los registros de memoria ubicados en el espacio C.
- C(put): Función para escribir los registros de memoria ubicados en el espacio C.
- CIO (get): Función para leer los registros de memoria ubicados en el espacio CIO.
- CIO (put): Función para escribir los registros de memoria ubicados en el espacio CIO.
- ClockRead: Lee el reloj del PLC.
- ClockWrite: Ajusta el reloj del PLC.
- Close PLC: Cierra un PLC abierto previamente.
- D (get): Función para leer los registros de memoria ubicados en el espacio D.
- D (put): Función para escribir los registros de memoria ubicados en el espacio D.
- DM (get): Función para leer los registros de memoria ubicados en el espacio DM.
- DownloadProgram: Descarga un programa para un PLC.
- DR (get): Función para leer los registros de memoria ubicados en el espacio DR.
- DR (put): Función para escribir los registros de memoria ubicados en el espacio DR.
- E (get): Función para leer los registros de memoria ubicados en el espacio E.

- E (put): Función para escribir los registros de memoria ubicados en el espacio E.
- EM (put): Función para leer los registros de memoria ubicados en el espacio EM.
- EM (get): Función para escribir los registros de memoria ubicados en el espacio EM.
- GetDeviceConfig: Esta función sirve para leer los parámetros de configuración del PLC.
- Help: Muestra información de ayuda.
- H (get): Función para leer los registros de memoria ubicados en el espacio H.
- H(put): Función para escribir los registros de memoria ubicados en el espacio H.
- HR (get): Función para leer los registros de memoria ubicados en el espacio HR.
- HR(put): Función para escribir los registros de memoria ubicados en el espacio HR.
- InitCXCerver: Inicializa el Cx-Server.
- IR (get): Función para leer los registros de memoria ubicados en el espacio IR.
- IR (put): Función para escribir los registros de memoria ubicados en el espacio IR.
- IsBadQuality: Comprueba si un punto determinado (o dirección) esta deteriorado.
- IsPointValid: Comprueba si un punto determinado (o dirección) existe o es válida.
- ListPoints: Devuelve la lista de puntos en un proyecto (o en un PLC).
- LR (get): Función para leer los registros de memoria ubicados en el espacio LR.
- LR (put): Función para escribir los registros de memoria ubicados en el espacio LR.
- OpenPLC: Abre un PLC para las comunicaciones.
- Protect: Protege la memoria de programa.
- RawFINS: Esta función permite enviar información a la dirección FINS indicada.
- Read: Lee el valor de un punto del PLC.

- ReadArea: Lee un bloque específico de la memoria de un PLC.
- RunMode (put): Escribe el modo de funcionamiento de un PLC.
- RunMode (get): Lee el modo de funcionamiento actual de un PLC.
- SetDefaultPLC: Esta función informa que PLC se ha establecido por defecto.
- SetDeviceAddress: Esta función puede utilizarse para establecer los elementos de una dirección de dispositivo (el número de red, el número de nodo, número de unidad y la dirección IP de Ethernet).
- SetDeviceConfig: Esta es una función se emplea para fijar cualquier elemento de CX-Server de configuración del PLC dispositivos.
- SR (get): Función para leer los registros de memoria ubicados en el espacio SR.
- SR (put): Función para escribir los registros de memoria ubicados en el espacio SR.
- ST (get): Función para leer los registros de memoria ubicados en el espacio ST.
- ST (put): Función para escribir los registros de memoria ubicados en el espacio ST.
- StopData: Función para detener Eventos.
- T (get): Función para leer los registros de memoria ubicados en el espacio T.
- T (put): Función para escribir los registros de memoria ubicados en el espacio T.
- TC (get): Función para leer los registros de memoria ubicados en el espacio TC.
- TC (put): Función para escribir los registros de memoria ubicados en el espacio TC.
- TCGetStatus: Esta función te devuelve los datos del estado del PLC.
- TCRemoteLocal: Esta función ejecutará el comando remoto / local para un PLC especifico.
- TK (get): Función para leer los registros de memoria ubicados en el espacio TK.
- TypeName: Función para leer el tipo de PLC.
- UploadProgram: Carga un programa desde un PLC.
- Value (get): Lee el valor de una dirección de un PLC. Esta función permite valores lógicos.

- Value (put): Escribe un valor a una dirección de un PLC. Esta función permite valores lógicos.

- Values (get): Lee una tabla de valores de un PLC. Esta función permite valores lógicos.

- Values (put): Escribe una tabla de valores de un PLC. Esta función permite valores lógicos.

- W (get): Función para leer los registros de memoria ubicados en el espacio W.

- W (put): Función para escribir los registros de memoria ubicados en el espacio W.

- Write: Escribe el valor de un punto del PLC.

- WriteArea: Escribe un bloque de memoria a una área específica en un PLC.

Anexo 2

Para estructurar mejor los enunciados de los ejercicios, los he ubicado en la carpeta "enunciados".

En esta carpeta, "enunciados", están todas las especificaciones para los objetivos que nos mandan en cada práctica.

También, se encuentra la solución de cada actividad en formato .DOC

Anexo 3

Dentro de cada carpeta donde esta localizada la práctica, también, se encuentra la resolución del los ejercicios en formato de programación CX-Programmer.